

Proceedings of the  
Twenty-First International  
Conference on  
Software Engineering &  
Knowledge Engineering

Boston, Massachusetts  
July 1-3, 2009



**SEKE** 2009

**PROCEEDINGS**

# **SEKE 2009**

**The 21<sup>st</sup> International Conference on  
Software Engineering &  
Knowledge Engineering**

**Sponsored by**

Knowledge Systems Institute Graduate School, USA

**Technical Program**

**July 1-3, 2009**

**Hyatt Harborside Hotel, Boston, Massachusetts, USA**

**Organized by**

Knowledge Systems Institute Graduate School

Copyright © 2009 by Knowledge Systems Institute Graduate School

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

ISBN 1-891706-24-1 (paper)

Additional Copies can be ordered from:  
Knowledge Systems Institute Graduate School  
3420 Main Street  
Skokie, IL 60076, USA  
Tel:+1-847-679-3135  
Fax:+1-847-679-3166  
Email:office@ksi.edu  
<http://www.ksi.edu>

Proceedings preparation, editing and printing are sponsored by  
Knowledge Systems Institute Graduate School

Printed by Knowledge Systems Institute Graduate School

# Foreword

On behalf of the Program Committee Co-Chairs, who are listed below, and the Program Committee of the 2009 International Conference on Software Engineering and Knowledge Engineering (SEKE-2009), it is an honor to welcome you to SEKE-2009 in Boston, MA. It has been my pleasure as Program Committee Chair to help organize this year's impressive scientific and technical program and the technical proceedings. The proceedings contain the papers selected for presentation at SEKE-2009. I hope these proceedings will serve as a valuable reference for the research community.

The International Conference on Software Engineering and Knowledge Engineering has entered its 21st year. For the past twenty years, the Conference on Software Engineering and Knowledge Engineering has provided a unique, centralized, forum for academic and industrial researchers and practitioners to discuss the application of either software engineering methods in knowledge engineering or knowledge-based techniques in software engineering. As our profession has been and still is evolving rapidly, SEKE has always been eager to capture new aspects in Software Engineering and Knowledge Engineering as well as to discuss consolidated special topics in depth in order to produce sustainable value for its attendees. Current trends are reflected in special sessions on topics like, e.g., SOA and SOA-based software-engineering, interoperability and the semantic web, without losing sight of still unsolved problems in more established fields like, e.g., software process models, requirements, agents and multi-agent systems, to name only a few. Preference is given to papers that emphasize the transference of methods between both engineering disciplines; however, outstanding papers on software engineering or knowledge engineering alone can also be found.

The SEKE-2009 Program Committee selected papers for publication in the proceedings and presentation at the Conference based upon a rigorous review process of the full papers. We received an overwhelming 226 submissions from many countries. The acceptance rate for full papers is 38% and for short papers is 23%. This year, authors from thirty-six countries including: Argentina, Australia, Austria, Bahrain, Brazil, Canada, China, Egypt, France, Germany, India, Iran, Iraq, Ireland, Italy, Japan, Jordan, Malta, Mexico, Myanmar, Netherlands, Pakistan, Peru, Romania, Saudi Arabia, Singapore, South Korea, Spain, Sweden, Switzerland, Taiwan, Tunisia, Turkey, United Kingdom, United States and Vietnam will present papers at the conference.

I appreciate having had the opportunity to serve as the Program Chair for this Conference, and am very grateful for the outstanding efforts provided by the Program Committee Co-Chairs, Dr. Jerry Gao (San Jose State University, USA) and Dr. Du Zhang (California State University, USA). The Program Committee members, the special session organizers and reviewers provided excellent support in promptly reviewing the manuscripts. I want to extend my sincere and deepest thanks to Dr. Daniel Beimbom and Dr. Masoud Sadjadi as the Publicity Co-Chairs, to Dr. Jose' Carlos Maldonado as the South America Liaison and, last but by no means least, to Dr. Taghi Khoshgoftaar for his help and counsel in numerous issues around organizing the conference program. My appreciation also goes to the keynote speakers for sharing their insights and experiences with the conference attendees. I am grateful to the authors and sessions chairs for their time and efforts to make SEKE-2009 a successful event. As always, Dr. S. K. Chang of the Knowledge Systems Institute, USA, provided excellent guidance throughout the effort. We all owe a special debt of gratitude to the efforts of Mr. Daniel Li, of the Knowledge Systems Institute. Without his help, the whole organization process would not have been possible in the narrow time frame if at all.

Finally, I truly hope that you will enjoy the technical program of SEKE-2009 as well as the social events and use this outstanding event to talk to old and new friends. Around the conference, we encourage you to explore and enjoy the attractions our host city, Boston, has to offer.

Guido Wirtz  
SEKE-2009 Program Chair

# **The 21<sup>st</sup> International Conference on Software Engineering & Knowledge Engineering (SEKE 2009)**

**July 1-3, 2009  
Hyatt Harborside Hotel, Boston, Massachusetts, USA**

## **Conference Organization**

### **Steering Committee Chair**

**Shi-Kuo Chang**, *University of Pittsburgh, USA*

### **Steering Committee**

**Vic Basili**, *University of Maryland, USA*

**Bruce Buchanan**, *University of Pittsburgh, USA*

**C. V. Ramamoorthy**, *University of California, Berkeley, USA*

### **Conference Chair**

**Taghi M. Khoshgoftaar**, *Florida Atlantic University, USA*

### **Program Chair**

**Guido Wirtz**, *Bamberg University, Germany*

### **Program Co-Chairs**

**Jerry Gao**, *San Jose State University, USA*

**Du Zhang**, *California State University, USA*

## Program Committee

- Alain Abran**, *Universite du Quebec, Canada*  
**Silvia Teresita Acuna**, *Universidad Autnoma De Madrid, Spain*  
**Taiseera Albalushi**, *Sultan Qaboos University, Oman*  
**Edward B. Allen**, *Mississippi State University, USA*  
**Doo-Hwan Bae**, *Computer Science Dept. KAIST, Korea*  
**Ebrahim Bagheri**, *University of New Brunswick, Canada*  
**Rami Bahsoon**, *University of Birmingham, UK*  
**Xiaoying Bai**, *Tsinghua University, China*  
**Maria Teresa Baldassarre**, *University of Bari, Italy*  
**Purushotham Bangalore**, *University of Alabama at Birmingham, USA*  
**Muhammad Ali Barbar**, *Lero Irish, Italy*  
**Emese Bari**, *eBay, USA*  
**Daniel Beimborn**, *Bamberg University, Germany*  
**Nicolas Belloir**, *University of Pau et des Pays de l'Adour, France*  
**Alessandro Bianchi**, *University of Bari, Italy*  
**Jim Bieman**, *Colorado State University, USA*  
**Danilo Caivano**, *University of Bari, Italy*  
**Gerardo Canfora**, *University of Sannio, Italy*  
**Joao W. Cangussu**, *University of Texas at Dallas, USA*  
**Giovanni Cantone**, *University of Rome Tor Vergata, Italy*  
**Jeffrey C. Carver**, *University of Alabama, USA*  
**Garcia-Castro**, *Technical University of Madrid, Spain*  
**Jaelson Castro**, *Universidade Federal de Pernambuco, Brazil*  
**Christine W. Chan**, *University of Regina, Canada*  
**Keith C.C. Chan**, *The Hong Kong Polytechnic University, Hong Kong*  
**W.K. Chan**, *City University of Hong Kong, Hong Kong*  
**Kuang-Nan Chang**, *Eastern Kentucky University, USA*  
**Ned Chapin**, *InfoSci Inc., USA*  
**Shu-Ching Chen**, *Florida International University, USA*  
**Yinong Chen**, *Arizona State University, USA*  
**Yoonsik Cheon**, *University of Texas at El Paso, USA*  
**Peter J. Clarke**, *Florida International University, USA*  
**Nelly Condori F.**, *Universidad Politecnica de Valencia, Spain*  
**Panos Constantopoulos**, *Athens University of Economics, Greece*  
**Dan Cooke**, *Texas Tech University, USA*  
**Kendra Cooper**, *University of Texas at Dallas, USA*  
**Maria Francesca Costabile**, *University of Bari, Italy*  
**Karl Cox**, *UNSW / Enterprise Analysts Pty Ltd, Australia*  
**Juan J. Cuadrado-Gallego**, *University of Alcala, Spain*  
**Alfredo Cuzzocrea**, *University of Calabria, Italy*  
**Deepak Dhungana**, *Johannes Kepler University, Austria*  
**Jin Song Dong**, *National University of Singapore, Singapore*  
**Jing Dong**, *University of Texas at Dallas, USA*  
**Dirk Draheim**, *University of Innsbruck, Austria*  
**Philippe Dugerdil**, *HEG-University of Applied Sciences, Switzerland*  
**Reiner Dumke**, *University of Magdeburh, Germany*  
**Schahram Dustdar**, *University of Technology Vienna, Austria*  
**Christof Ebert**, *Vector Consulting Services GmbH, Germany*  
**Raimund K. Ege**, *Northern Illinois University, USA*

**Faezeh Ensan**, *University of New Brunswick, Canada*  
**Onyeka Ezenwoye**, *South Dakota State University, USA*  
**Davide Falessi**, *Universita degli Studi di Roma Tor Vergata, Italy*  
**Behrouz Homayoun Far**, *University of Calgary, Canada*  
**Robert Feldt**, *Blekinge Institute of Technology, Sweden*  
**Eduardo B. Fernandez**, *Florida Atlantic University, USA*  
**Renata Fortes**, *University of Sao Paulo, Brazil*  
**Jerry Gao**, *San Jose State University, USA*  
**Kehan Gao**, *Eastern Connecticut State University, USA*  
**Alessandro Garcia**, *Lancaster University, UK*  
**Felix Garcia**, *University of Castilla-La Mancha, Spain*  
**Holger Giese**, *Hasso Plattner Institut, Germany*  
**Itana Gimenes**, *UEM/PR/Brazil, Brazil*  
**Swapna Gokhale**, *University of Connecticut, USA*  
**Wolfgang Golubski**, *Zwickau University of Applied Sciences, Germany*  
**Des Greer**, *Queens University Belfast, UK*  
**Eric Gregoire**, *Universite dArtois, France*  
**Mark Harman**, *Kings College London, UK*  
**Xudong He**, *Florida International University, USA*  
**Rattikorn Hewett**, *Texas Tech University, USA*  
**Mei Hsing**, *Fu Jen Catholic University, Taiwan*  
**Shihong Huang**, *Florida Atlantic University, USA*  
**Byung-Yeon Hwang**, *The Catholic University of Korea, Korea*  
**Ali Idri**, *ENSIAS, Rabat, Morocco*  
**Peter In**, *Korea University, Korea*  
**Clinton Jeffery**, *University of Idaho, USA*  
**Natalia Juristo**, *Madrid Technological University, Spain*  
**Audris Kalnins**, *University of Latvia, Latvia*  
**Taghi Khoshgoftaar**, *Florida Atlantic University, USA*  
**Sascha Konrad**, *Siemens Corporate Research, USA*  
**Gunes Koru**, *University of Maryland, Balt Cty, USA*  
**Nicholas A. Kraft**, *University of Alabama, USA*  
**Vinay Kulkarni**, *Tata Research, India*  
**Gi-Hwon Kwon**, *College of Natural Sciences, Korea*  
**Mark Last**, *Ben-Gurion University of the Negev, Israel*  
**Konstantin Laufer**, *Loyola University Chicago, USA*  
**Jeff Lei**, *University of Texas at Arlington, USA*  
**Tao Li**, *Florida International University, USA*  
**Shih-Hsi Liu**, *California State University at Fresno, USA*  
**Xiaodong Liu**, *Napier University, UK*  
**Yan (Jenny) Liu**, *National ICT, Australia*  
**Yi Liu**, *Georgia college and State University, USA*  
**Hakim Lounis**, *University of Quebec, CA*  
**Zhongyu (Joan) Lu**, *The University of Huddersfield, UK*  
**Heiko Ludwig**, *IBM TJ Watson Research Center, Almaden, San Jose, USA*  
**Michael R. Lyu**, *Chinese University of Hong Kong, Hong Kong*  
**Jose Carlos Maldonado**, *University of Sao Paulo, Brazil*  
**Antonio Mana**, *University of Malaga, Spain*  
**Emilia Mendes**, *University of Auckland, New Zealand*  
**Harald Meyer**, *HPI Potsdam, Germany*  
**Rym Mili**, *University of Texas at Dallas, USA*

**James Miller**, *University of Alberta, Canada*  
**Ana M. Moreno**, *Techincal University of Madrid, Spain*  
**Henry Muccini**, *Univerita degli Studi de L Aquila, Italy*  
**Martin Neil**, *MQueen Mary (U. of London), UK*  
**Allen Nikora**, *Jet Propulsion Laboratory, USA*  
**Elisabetta Di Nitto**, *Politecnico de Milano, Italy*  
**Mehmet Orgun**, *Macquarie University, Australia*  
**Eric Pardede**, *La Trobe University, Australia*  
**Witold Pedrycz**, *University of Alberta, Canada*  
**Jun Peng**, *Chongqing University of Science and technology, China*  
**Massimiliano Di Penta**, *University of Sannio, Italy*  
**Antonio Piccinno**, *University of Bari, Italy*  
**Alfonso Pierantonio**, *University of L.Aquila, Italy*  
**Damith C. Rajapakse**, *National Univ. of Singapore, Singapore*  
**Rajeev Raje**, *Indiana University Purdue University, Indianapolis, USA*  
**Sanjay Ranka**, *University of Florida, USA*  
**Marek Refomat**, *University of Alberta, Canada*  
**Marek Reformat**, *University of Alberta, Canada*  
**Robert Reynolds**, *Wayne State University, USA*  
**Daniel Rodriguez**, *The University of Alcalá, Spain*  
**Ignacio Garcia Rodriguez**, *Universidad de Castilla-La Mancha, Spain*  
**Guenther Ruhe**, *University of Calgary, Canada*  
**Samira Sadaoui**, *University of Regina, Canada*  
**Masoud Sadjadi**, *Florida International University, USA*  
**Eng. Sattar B. Sadkhan**, *University of Babylon, Iraq*  
**Ramon Sagarna**, *The University of Birmingham, UK*  
**Ahmed Salem**, *California State University at Sacramento State, USA*  
**S. Alessandro Sarcia**, *Universita degli Studi di Roma Tor Vergata, Italy*  
**Kamran Sartipi**, *McMaster University, Canada*  
**Peter Sawyer**, *University of Lancaster, UK*  
**Douglas Schmidt**, *Vanderbilt University, USA*  
**Naeem Seliya**, *University of Michigan at Dearborn, USA*  
**Tony Shan**, *IBM, USA*  
**Rajan Shankaran**, *Macquarie University, Australia*  
**Yidong Shen**, *Chinese Academy of Science, China*  
**Michael Shin**, *Texas Tech University, USA*  
**George Spanoudakis**, *City University, UK*  
**Arndt von Staa**, *PUC-Rio, Brazil*  
**Rajesh Subramanyan**, *Siemens Corporate Research, Inc. USA*  
**Jeff Tian**, *Southern Methodist University, USA*  
**Scott Tilley**, *Florida Institute of Technology, USA*  
**Mark Trakhtenbrot**, *Holon Institute of Technology, Israel*  
**Laurence Tratt**, *Bournemouth University, UK*  
**Peter Troger**, *Blekinge Institute of Technology, Sweden*  
**Jeffrey Tsai**, *University of Illinois, USA*  
**Tse-Ming Tsai**, *Institute for Information Industry, Taiwan*  
**T.H. Tse**, *University of Hong Kong, Hong Kong*  
**Antonio Vallecillo**, *University of Malaga, Spain*  
**Michael VanHilst**, *Florida Atlantic University, USA*  
**Sira Vegas**, *Universidad Politecnica de Madrid, Spain*  
**Silvia Regina Vergilio**, *UFPR, Brazil*



**Huanjing Wang**, *Western Kentucky University, USA*  
**Christiane Gresse von Wangenheim**, *Universidade do Vale do Itaja, Brazil*  
**Tim Weitzel**, *Bamberg University, Germany*  
**Victor Winter**, *University of Nebraska at Omaha, USA*  
**Guido Wirtz**, *Bamberg University, Germany*  
**Eric Wong**, *University of Texas at Dallas, USA*  
**Franz Wotawa**, *Technische Universitaet Graz, Austria*  
**Haiping Xu**, *University of Massachusetts Dartmouth, USA*  
**Chi-Lu Yang**, *Institute for Information Industry, Taiwan*  
**Hongji Yang**, *De Montfort University, UK*  
**Ren-Dar Yang**, *Institute for Information Industry, Taiwan*  
**Huiqun Yu**, *East China University of Science and Technology, China*  
**Du Zhang**, *California State University Sacramento, USA*  
**Jing Zhang**, *Motorola Research, USA*  
**Min-Ling Zhang**, *Hohai University, China*  
**Zhinan Zhou**, *Samsung, USA*  
**Hong Zhu**, *Oxford Brookes University, UK*  
**Xingquan Zhu**, *Florida Atlantic University, USA*  
**Eugenio Zimeo**, *University of Sannio, Italy*  
**Andrea Zisman**, *City University, UK*

## **PUBLICITY CO-CHAIRS**

**Daniel Beimborn**, *Bamberg University, Germany*  
**S. Masoud Sadjadi**, *Florida International University, USA*

## **South America Liasion**

**Jose Carlos Maldonado**, *University of Sao Paulo, Brazil*

## **Industry Advisory Committee**

**Yi Deng**, *Dean, School of Computer Science, Florida International University, USA*  
**J. S. Ke**, *Senior Fellow, Institute for Information Industry, Taiwan*  
**A. J. Rhem**, *Senior Partner, A. J. Rhem and Associates Inc., USA*

## **Proceedings Cover Design**

**Gabriel Smith**, *Knowledge Systems Institute Graduate School, USA*

## **Conference Secretariat**

**Judy Pan**, *Chair, Knowledge Systems Institute Graduate School, USA*  
**Omasan Etuwewe**, *Knowledge Systems Institute Graduate School, USA*  
**Chen-Cheang Huang**, *Knowledge Systems Institute Graduate School, USA*  
**Daniel Li**, *Knowledge Systems Institute Graduate School, USA*

# Table of Contents

<b>Foreword .....</b>	<b>iii</b>
<b>Conference Organization .....</b>	<b>iv</b>
<b>Wireless Computing, Networking and Sensing</b>	
<i>Dr. H. T. Kung .....</i>	<b>1</b>
<b>Virtual Spaces: From the Past to the Future</b>	
<i>Dr. Shi-Kuo Chang .....</i>	<b>2</b>
<b>Software Engineering of Autonomic Grid Computing Systems and Applications</b>	
Web Services Reliability Patterns (S)	
<i>Ingrid Buckley, Eduardo B. Fernandez, Gustavo Rossi, S. Masoud Sadjadi .....</i>	<b>4</b>
Consistency in Self-Reconfiguration of Self-Healing Systems	
<i>Michael E. Shin, Kiran Gopala Reddy Sunanda .....</i>	<b>10</b>
Task Decomposition for Adaptive Data Staging in Workflows for Distributed Environments (S)	
<i>Onyeka Ezenwoye, Balaji Viswanathan, S. Masoud Sadjadi, Liana Fong, Gargi Dasgupta, Selim Kalayci .....</i>	<b>16</b>
<b>Requirements</b>	
Constructing FODA Feature Diagrams with a GUI-based Tool (S)	
<i>Shin Nakajima .....</i>	<b>20</b>

Towards a Classification of Requirements Relationships <i>Ruhaya Ab Aziz, Didar Zowghi, Tom McBride</i> .....	26
Towards the Selection of the Most Suitable Elicitation Technique Through a Defined Requirements Elicitation Process (S) <i>Marcelo Werneck Barbosa, Glivia Angelica Rodrigues Barbosa</i> .....	33
A Requirement Traceability Refinement Method Based on Relevance Feedback <i>Lingjun Kong, Juan Li, Yin Li, Ye Yang, Qing Wang</i> .....	37
Applying Transformation Rules to Improve i* Models (S) <i>Marcia Lucena, Carla Silva, Emanuel Santos, Fernanda Alencar, Jaelson Castro</i> .....	43
Nested NL Representation for OO Analysis and Design (S) <i>Magda G. Ilieva, Olga Ormandjieva</i> .....	49
Specification of Data Requirements from Task Descriptions <i>Jose Luis de la Vara, Juan Sanchez</i> .....	55
From Organizational Models to Software Requirements <i>Alicia Martinez, Oscar Pastor, John Mylopoulos, Hugo Estrada</i> .....	61
Systematic Review of Requirements Reuse <i>Flavia Braga de Azambuja, Ricardo Melo Bastos, Ana Paula Terra Bacelo</i> .....	67
Reprioritizing the Requirements in Agile Software Development: Towards a Conceptual Model from Clients' Perspective <i>Zornitza Racheva, Maya Daneva</i> .....	73
<b>Data Mining and Features</b>	
A Novel Hybrid Search Algorithm for Feature Selection <i>Pengpeng Lin, Huanjing Wang, Taghi M. Khoshgoftaar</i> .....	81
Improving Text Document Clustering by Exploiting Open Web Directory <i>Gaurav Ruhela, P.Krishna Reddy</i> .....	87

Automated Nursing Knowledge Management Using Indexing (S) <i>Shihong Huang, Sucharita Chinchankar, Abhijit Pandya, Sam Hsu, Marilyn Parker</i>	93
---	----

## Applications

Classifying Web Robots by K-means Clustering <i>Derek Doran, Swapna S. Gokhale</i>	97
---	----

Systematic Risk Assessment and Cost Estimation for Software Problems <i>Jerry Gao, Maulik Shah, Mihir Shah, Devarshi Vyas, Pushkala Pattabhiraman, Kamini Dandapani, Emese Bari</i>	103
--	-----

Improving Negotiations through Fuzzy Cognitive Maps <i>Sergio Assis Rodrigues, Tiago Santos da Silva, Jano Moreira de Souza</i>	110
--	-----

## Software Engineering with Computational Intelligence and Machine Learning

Value-Based Software Quality Modeling <i>Naeem Seliya, Taghi M. Khoshgoftaar</i>	116
---	-----

Predicting Maintainability expressed as Change Impact: A Machine-learning-based Approach <i>H. Lounis, M.K. Abdi, H. Sahraoui</i>	122
--	-----

Program File Bug Fix Effort Estimation Using Machine Learning Methods for OSS (S) <i>Syed Nadeem Ahsan, Javed Ferzund, Franz Wotawa</i>	129
--	-----

## Software Architecture and Evolution

An Architecture-based Evolution Management Method for Software Product Line <i>Xin Peng, Liwei Shen, Wenyun Zhao</i>	135
---	-----

Towards Design and Architectural Evaluation of Product Variants: A Case Study on an Open Source Software System <i>Muhammad Irfan Ullah, Guenther Ruhe, Vahid Garousi</i>	141
--	-----

Decision Support System Environment for Software Architecture Style Selection (DESAS v1.0) (S) <i>Shahrouz Moaven, Hamed Ahmadi, Jafar Habibi, Ali Kamandi</i> .....	147
Towards Architecture-centric Collaborative Software Development (S) <i>Yanchun Sun, Hui Song, Wenpin Jiao</i> .....	152
<b>Agents and Multi-Agent Systems</b>	
Analysis of Agent Oriented Software Engineering Methodologies for Social Causal Models <i>Michele Atkinson, Sheryl Duggins</i> .....	157
Realization of Semantic Search Using Concept Learning and Document Annotation Agents <i>Behrouz H. Far, Cheng Zhong, Zilan Yang, Mohsen Afsharchi</i> .....	164
Agent-based Simulation Model for the Evolution Process of Open Source Software <i>Taemin Seo, Heesang Lee</i> .....	170
Towards Merging Goal Models of Networked Software <i>Zaiwen Feng, Keqing He, Rong Peng, Jian Wang, Yutao Ma</i> .....	178
Comparison of Some Single-agent and Multi-agent Information Filtering Systems on a Benchmark Text Data Set (S) <i>Snehasis Mukhopadhyay, Shengquan Peng, Rajeev Raje, Mathew Palakal, Javed Mostafa</i> .....	185
Towards Adaptable BDI Agent: A Formal Aspect-oriented Modeling Approach (S) <i>Lily Chang, Xudong He</i> .....	189
A Multi-agent Debugging Extension Architecture (S) <i>Ziad Al-Sharif, Clinton Jeffery</i> .....	194
A Recognition-primed Architecture for Human-centric Multi-agent Systems <i>Xiacong Fan</i> .....	200

Using Knowledge Objects to Exchange Knowledge in a MAS Platform <i>Ana Paula Lemke, Marcelo Blois</i> .....	206
JAAF: A Framework to Implement Self-adaptive Agents <i>Baldoino F. dos S. Neto, Andrew D. da Costa, Manoel T. de A. Netto, Viviane T. da Silva, Carlos J. P. de Lucena</i> .....	212
An Agent-based Centralized e-Marketplace in a Virtual Environment (S) <i>Ingo Seidel, Markus Gartner, Josef Froschauer, Helmut Berger, Dieter Merkl</i> .....	218
<b>Interoperability and Semantic Web Technologies</b>	
Semantic Service Matchmaking in the ATM Domain Considering Infrastructure Capability Constraints (S) <i>Thomas Moser, Richard Mordinyi, Wikan Danar Sunindyo, Stefan Biffl</i> .....	222
Ontology Mapping Representations: A Pragmatic Evaluation (S) <i>Hendrik Thomas, Declan O'Sullivan, Rob Brennan</i> .....	228
Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas <i>Stefan Biffl, Wikan Danar Sunindyo, Thomas Moser</i> .....	233
LD2SD: Linked Data Driven Software Development <i>Aftab Iqbal, Oana Ureche, Michael Hausenblas, Giovanni Tummarello</i> .....	240
Improving Searchability of a Music Digital Library with Semantic Web Technologies <i>Paloma de Juan, Carlos A. Iglesias</i> .....	246
A Guideline Engine For Knowledge Management in Clinical Decision Support Systems (CDSSs) <i>Michele Ceccarelli, Alessandro De Stasio, Antonio Donatiello, Dante Vitale</i> .....	252
Ontology-based Semantic Annotations of Medical Articles (S) <i>Jihen Majdoubi, Mohamed Tmar, Faiez Gargouri</i> .....	258

## Reverse Engineering

Automating Business Intelligence Recovery from a Web-based System  
*Jian Kang, Jianzhi Li, Jianchu Huang, Yingchun Tian, Hongji Yang* ..... 262

Automatic Class Matching to Compare Extracted Class Diagrams: Approach and Case Study (S)  
*Yan Liang, Nicholas A. Kraft, Randy K. Smith* ..... 268

## SOA-based software engineering

Modeling and Verification of Automatic Multi-business Transactions  
*Min Yuan, Zhiqiu Huang, Jian Zhao, Xiang Li* ..... 274

An Adaptive Management Framework for Service Brokers in Service-oriented Architecture  
*W.T. Tsai, Tszyan Chow, Yinong Chen, Xiao Wei* ..... 280

Requirements Discovery Based on RGPS Using Evolutionary Algorithm (S)  
*Tao Peng, Bing Li, Weifeng Pan, Zaiwen Feng* ..... 286

Mediation Based Variability Modeling for Service Oriented Software Product Lines (S)  
*Mohammad Abu-Matar* ..... 291

## Software Engineering Education

Pedagogy-oriented Software Modeling and Simulation of Component-based Physical Systems  
*Dan Tappan* ..... 295

An Academia-Industry Collaborative Teaching and Learning Model for Software Engineering Education (S)  
*Huilin Ye* ..... 301

## Software Testing and Automation

Data Flow Analysis and Testing for Web Service Compositions Based on WS-BPEL <i>Chien-Hung Liu, Shu-Ling Chen</i> .....	306
Knowledge-based Software Test Generation <i>Valeh H. Nasser, Weichang Du, Dawn MacIsaac</i> .....	312
Some Experiments on Test Case Traceability (S) <i>Macario Polo, Beatriz Perez, Pedro Reales</i> .....	318
<b>Service Oriented Architecture</b>	
Business Modeling for Service Engineering: Toward an integrated Procedure Model <i>Gregor Scheithauer, Stefan Augustin, Guido Wirtz</i> .....	322
A Systematic SOA-based Architecture Process <i>Jose Jorge Lima Dias Junior, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira</i> .....	328
Research and Implementation of Service-oriented Architecture Supporting Location-based Services on Sensor Networks (S) <i>Bin-Yi Liao, Wen-Shyang Huang, Jeng-Shyang Pan, Hong-Chi Wu, Yuh-Ming Cheng, Jen-Kuin Lee, Bo-Sian Wang, E-Liang Chen, Mong-Fong Horng</i> .....	334
Service Creation and Composition for Realization On Service-oriented Architecture <i>Chi-Lu Yang, Yeim-Kuan Chang, Chih-Ping Chu</i> .....	338
An Extendible Translation of BPEL to a Machine-verifiable Model <i>John C. Sloan, Taghi M. Khoshgoftaar, Augusto Varas</i> .....	344
Generating Test Cases of Composite Services Based on OWL-S and EH-CPN <i>Bixin Li, Ju Cai, Dong Qiu, Shunhui Ji, Yuting Jiang</i> .....	350



User Perceived Response-time Optimization Method for Composite Web Services <i>Junfeng Zhao, Yasha Wang, Bing Xie</i> .....	356
Dynamic Service Composition for Virtual UPnP Device Creation <i>Sheng-Tzong Cheng, Chih-Lun Chou, Jiashing Shih, Mingzoo Wu</i> .....	364
Using Service-oriented Architectures for Socio-Cultural Analysis <i>David Garlan, Kathleen M. Carley, Bradley Schmerl, Michael Bigrigg, Orieta Celiku</i> .	370
<b>Languages and Program Understanding</b>	
A Conceptual Model for Comprehension of Object-oriented Interactive Systems (S) <i>Izuru Kume, Etsuya Shibayama</i> .....	376
Arabic Lisp (S) <i>Hanan Elazhary</i> .....	382
The Use of Reading Technique and Visualization for Program Understanding <i>Daniel Porto, Manoel Mendonca, Sandra Fabbri</i> .....	386
Language Support for Event-based Debugging <i>Ziad Al-Sharif, Clinton Jeffery</i> .....	392
Pie Tree Visualization <i>Mireille Samia, Michael Leuschel</i> .....	400
Formal Verification of Scalable NonZero Indicators <i>Shao Jie Zhang, Yang Liu, Jun Sun, Jin Song Dong, Wei Chen, Yanhong A. Liu</i> ...	406
<b>Software Quality</b>	
Detecting Defects with an Interactive Code Review Tool Based on Visualisation and Machine Learning <i>Stefan Axelsson, Dejan Baca, Robert Feldt, Darius Sidlauskas, Denis Kacan</i> .....	412

Dynamic Test Profiles in Adaptive Random Testing: A Case Study (S) <i>Huai Liu, Fei-Ching Kuo, Tsong Yueh Chen</i> .....	418
A Novel Method of Mutation Clustering Based on Domain Analysis (S) <i>Changbin Ji, Zhenyu Chen, Baowen Xu, Zhihong Zhao</i> .....	422
Using a Mining Frequency Patterns Model to Automate Passive Testing of Real-time Systems <i>Cesar Andres, Mercedes G. Merayo, Manuel Nunez</i> .....	426
DeLLIS: A Data Mining Process for Fault Localization (S) <i>Peggy Cellier, Mireille Ducasse, Sebastien Ferre, Olivier Ridoux</i> .....	432
Extending AOP to Support Broad Runtime Monitoring Needs (S) <i>Amjad Nusayr, Jonathan Cook</i> .....	438
Clustering of Defect Reports Using Graph Partitioning Algorithms (S) <i>Vasile Rus, Xiaofei Nan, Sajjan Shiva, Yixin Chen</i> .....	442
Documenting Quality Attributes of Software Components (S) <i>Wenhui Zhu, Yanchun Sun, Gang Huang, Hong Mei</i> .....	446
Taming Inconsistency in Value-based Software Development <i>Du Zhang</i> .....	450
WSTester: Testing Web Service for Behavior Conformance (S) <i>Bixin Li, Lili Yang, Shunhui Ji, Dong Qiu, Xufang Gong</i> .....	456
Robustness Verification Challenges in Automotive Telematics Software (S) <i>Ali Shahrokni, Robert Feldt, Fredrik Petterson, Anders Back</i> .....	460
 <b>Smart Environments and Applications</b>	
A 2D-barcode Based Mobile Advertising Solution <i>Jerry Zeyu Gao, Hema Veeraragavathatham, Shailashree Savanur, Jinchun Xia</i> .....	466

Long-term Prediction of Wireless Network Traffic <i>Zhiwei Xu, Zhou Zhou, Weibiao Wu</i> .....	473
<b>Software Architecture and Applications</b>	
Resource Allocation for a Modular Software System <i>Lance Fiondella, Swapna S. Gokhale</i> .....	480
Enhancing Property Specification Tools With Validation Techniques <i>Salamah Salamah, Matthew Del Buono, Eric Baily, Sarah Printy, Derek Ferris, Laurel Christian</i> .....	487
<b>Software Process and Process Models</b>	
Supporting Good Decision Making at Early Stage of Software Design <i>Hung-Fu Chang, Stephen C-Y. Lu</i> .....	493
A Language for Modeling Software Development Life Cycles (S) <i>Ernest Cachia, Mark Micallef</i> .....	499
Weaving Process Patterns into Software Process Models (S) <i>Xiao-yang He, Ya-sha Wang, Jin-gang Guo, Wu Zhou, Jia-kuan Ma</i> .....	505
Assessing Workflow Ability of ERP and WfM Systems (S) <i>Lerina Aversano, Roberto Intonti, Maria Tortorella</i> .....	509
Mining Objective Process Metrics from Repository Data <i>Michael VanHilst, Shihong Huang</i> .....	514
Collaborative Development of System Architecture - a Tool for Coping with Inconsistency (S) <i>Peter Henderson, Matthew J. Henderson</i> .....	520
BITS: Issue Tracking and Project Management Tool in Healthcare Software Development (S) <i>Ayşe Tosun, Ayşe Bener, Ekrem Kocaguneli</i> .....	526

## Security and Privacy

Privacy-preserving Clustering of Data Streams  
*Ching-Ming Chao, Chih-Chin Shen* ..... 530

FiGD: An Open Source Intellectual Property Violation Detector (S)  
*Carson Brown, David Barrera, Dwight Deugo* ..... 536

Integrating Privacy Requirements into Security Requirements Engineering  
*Saeed Abu-Nimeh, Seiya Miyazaki, Nancy R. Mead* ..... 542

iPass: An Integrated Framework for Educating, Monitoring and Enforcing Password Policies for Online Services (S)  
*Dhananjay Kulkarni, Diana Ciric, Fernanda Zulkarnain* ..... 548

## Ontologies and their Applications

Improving Natural Language Specifications with Ontologies  
*Sven J. Korner, Torben Brumm*..... 552

A Knowledge-based Retrieval Model  
*Fabio Silva, Rosario Girardi, Lucas Drumond* ..... 558

TRIPLE Content-based Ontology (TRICOT) for XML Dissemination (S)  
*Mirella M. Moro, Deise de Brum Saccol, Renata de Matos Galante* ..... 564

An Ontology-based Approach to Portable Embedded System Development  
*Feng Chen, Hong Zhou, Jianzhi Li, Ruimin Liu, Hongji Yang, Han Li, He Guo, Yuxin Wang* ..... 569

An Integrated Ontology Framework for Health Information Exchange  
*S. Demurjian, R. Saripalle, S. Berhe* ..... 575

## HCI and Smart Environments

Modeling User Interpersonal Stances in Affective Dialogues with an ECA <i>Nicole Novielli, Enrica Gentile</i> .....	581
Capturing Users' Preferences and Intentions in a Semantic Search System (S) <i>Caio Stein D'Agostini, Renato Fileto</i> .....	587
Toward Developing Knowledge Representation in Emergency Medical Assistance through a Ontology-based Semantic Cache Model (S) <i>Heloise Manica, Cristiano C. da Rocha, Jose Leomar Todesco, M. A. R. Dantas, Michael A. Bauer</i> .....	592
Specification of a Component-based Domotic System to Support User-defined Scenarios <i>Fady Hamoui, Marianne Huchard, Christelle Urtado, Sylvain Vauttier</i> .....	597
Towards Mobility Support in Smart Environments (S) <i>Daniel Retkowitz, Ibrahim Armac, Manfred Nagl</i> .....	603
A Graph Transformation-based Approach to Task Allocation in Wireless Sensor Actor Networks (S) <i>Hossein Momeni, Vahid Rafe, Mohsen Sharifi, Adel T. Rahmani</i> .....	609
<b>Software Measurement</b>	
Another New Criterion to Improve the Interaction Diagrams Quality <i>Lilia Grati, Mohamed Tmar, Faiez Gargouri</i> .....	613
Software Project Effort Estimation Non Lineal Mathematical Models <i>Pablo R. Soria, Borja Martin, Marian Fernandez de Sevilla, Maria J. Dominguez- Alda, Miguel A. Herranz</i> .....	619
Software Estimation: Universal Models or Multiple Models? (S) <i>Alain Abran, Juan Jose Cuadrado Gallego</i> .....	625

An Empirical Study of the Feedback of the In-process Measurement in a Japanese Consortium-type Software Project (S) <i>Yoshiki Mitani, Tomoko Matsumura, Katsuro Inoue, Mike Barker, Akito Monden, Ken-ichi Matsumoto</i> .....	631
Prest: An Intelligent Software Metrics Extraction, Analysis and Defect Prediction Tool <i>Ekrem Kocaguneli, Ayse Tosun, Ayse Bener, Burak Turhan, Bora Caglayan</i> .....	637
Accelerated Risk Management using Statistical Triggers <i>Rose Williams, Krishna Ratakonda</i> .....	643
 <b>Process Management and Outsourcing</b>	
A Layered Approach for Planning Releases under Uncertain Capacities <i>Jim McElroy, Guenther Ruhe</i> .....	649
PP-HAS: A Task Priority Based Preemptive Human Resource Scheduling Method <i>Lizi Xie, Qing Wang, Junchao Xiao, Yongji Wang, Ye Yang</i> .....	655
A Real Execution of a Software Process Improvement: An Opportunity to Execute a Combination of Approaches (S) <i>Adriano Bessa Albuquerque, Ana Regina Rocha</i> .....	661
Establish Decision Making Process for Selecting Outsourcing Company <i>Akihiro Hayshi</i> .....	666
From Strategy to Solution: A Lightweight Semi-prescriptive Approach for Software Development Lifecycle with Outsourcing Support (S) <i>Nelio Alves, Sergio Paim, Alexandre Cardoso, Edgard Lamounier</i> .....	672
 <b>Databases and Data Modeling</b>	
A Model Driven Method for Data Warehouse <i>Leopoldo Zepeda, Elizabeth Cecena, Jorge Rivas, Javier Cano, Nelly Condory, Matilde Celma</i> .....	676

Analyzing the Software Development Process with SyQL and Lagrein <i>Mirco Bianco, Alberto Sillitti, Giancarlo Succi</i> .....	682
Performance Analysis of a Deductive Database with a Semantic Web Reasoning Engine: ConceptBase and Racer <i>Simone A. Ludwig, Craig Thompson, Kristofor Amundson</i> .....	688
Object Specification Language for Graph Based Conceptual level Multidimensional Data Model (S) <i>Anirban Sarkar, Sankhayan Choudhury, Nabendu Chaki, Swapan Bhattacharya</i> .....	694
A Framework for Trajectory Data Preprocessing for Data Mining (S) <i>Luis Otavio Alvares, Gabriel Oliveira, Carlos A. Heuser, Vania Bogorny</i> .....	698
<b>Multimedia Software Engineering</b>	
A Payload Optimization Technique for Multimedia Visual Cryptography <i>Moussa H. Abdallah, Rola I. Al-Khalid, Randa A. Al-Dallah</i> .....	703
Knowledge Management Framework for Conference Video-recording Retrieval <i>Maria Sokhn, Elena Mugellini, Omar Abou Khaled</i> .....	709
<b>Software Engineering and Aspects</b>	
Separating The Scattered Concerns: A Graph Based Model <i>Dipankar Majumdar, Swapan Bhattacharya</i> .....	715
Early Analysis of Modularity in Software Product Lines <i>Jose M. Conejero, Juan Hernandez, Elena Jurado, Pedro J. Clemente, Roberto Rodriguez</i> .....	721
<b>Model-Driven Software Engineering</b>	
MD-JPA Profile: A Model Driven Language for Java Persistence <i>Alexandre Torres, Renata Galante, Marcelo S. Pimenta</i> .....	727

A Pragmatic UML-based Meta Model for Object-oriented Code Generation <i>Tobias Haubold, Georg Beier, Wolfgang Golubski</i> .....	733
An Ontology-based Model Driven Approach for a Music Learning System <i>Yingchun Tian, Feng Chen and Hongji Yang, Leigh Landy</i> .....	739
<b>Reviewer's Index</b> .....	745
<b>Author's Index</b> .....	748

**Note: (S) means short paper.**





# **Keynote I: Wireless Computing, Networking and Sensing**

**H. T. Kung**

Harvard School of Engineering and Applied Sciences  
Cambridge, MA

## **Abstract**

In the next decade we will begin to face very large sensor-generated datasets on the order of zettabytes or even yottabytes. While the bulk of the processing and storage must be distributed near the sensors, centralized control and applications could still be needed. Moreover, for flexibility in sensor deployments, communications over wireless networks will be essential in spite of their modest bandwidths. This talk will discuss fundamental challenges and recent research progress at Harvard in these areas.

## **About Dr. H. T. Kung**

H. T. Kung received his B.S. from National Tsing Hua University (Taiwan), and Ph.D. from Carnegie Mellon University. He is currently William H. Gates Professor of Computer Science and Electrical Engineering at Harvard University. Prior to joining Harvard in 1992, he taught at Carnegie Mellon for about eighteen years. Dr. Kung has pursued a variety of research interests, including complexity theory, database systems, VLSI design, parallel computing, computer architectures, computer networks, network security, wireless communications, and networking of unmanned aerial vehicles. He maintains a strong linkage with industry and has served as a consultant and board member to numerous organizations. Dr. Kung's professional honors include Member of the National Academy of Engineering in USA and Member of the Academia Sinica in Taiwan.

# **Keynote II: Virtual Spaces: From the Past to the Future**

**Shi-Kuo Chang**

## **Abstract**

Space can be seen in many different ways. When an architect and a computer scientist look at space they see very different things and yet sometimes they make surprisingly similar discoveries. As a computer scientist with strong research interests in visual languages I learned many things from the theory and practice of architecture. This lecture on virtual spaces is motivated by a desire to share these findings. We begin by discussing the origins of architectural pleasure and how the space of a dwelling can be divided into refuge and prospect according to Grant Hildebrand. This decomposition of space leads us naturally to consider spatial relations and patterns. On the pragmatic side we illustrate patterns by the works of Frank Lloyd Wright. On the theoretical side we consider Christopher Alexander's theory of patterns and its relationship to the theory of visual languages and software engineering. After a discussion of William Mitchell's e-topia as an example of the V-topia, the virtual cities of the past, the present and the future are surveyed.

## **About Dr. Laura Haas**

Dr. Chang received the B.S.E.E. degree from National Taiwan University in 1965. He received the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1967 and 1969, respectively. He was a research scientist at IBM Watson Research Center from 1969 to 1975. From 1975 to 1982 he was Associate Professor and then Professor at the Department of Information Engineering, University of Illinois at Chicago. From 1982 to 1986 he was Professor and Chairman of the Department of Electrical and Computer Engineering, Illinois Institute of Technology. From 1986 to 1991 he was Professor and Chairman of the Department of Computer Science, University of Pittsburgh. He is currently Professor and Director of the Center for Parallel, Distributed and Intelligent Systems, University of Pittsburgh. Dr. Chang is a Fellow of IEEE. He published over 230 papers and 16 scientific books. He is the founder and co-editor-in-chief of the international journal, Visual Languages and Computing, published by Academic Press, the editor-in-chief of the international journal, Software Engineering & Knowledge Engineering, published by World Scientific Press, and the co-editor-in-chief of the international journal on Distance Education Technologies. Dr. Chang pioneered the development of Chinese language computers, and was the first to develop a picture grammar for Chinese ideographs, and invented the phonetic phrase Chinese input method.

Dr. Chang's literary activities include the writing of over thirty novels, collections of short stories and essays. He is widely regarded as an acclaimed novelist in Taiwan. His novel, The Chess King, was translated into English and German, made into a stage musical, then a TV mini-series and a movie. It was adopted as textbook for foreign students studying Chinese at the Stanford Center (Inter-University Program for Chinese Language Studies administered by Stanford University), Taipei, Taiwan. In 1992, Chess King was adopted as supplementary reading for high school students in Hong Kong. The short story, "Banana Boat", was included in a textbook for advanced study of Chinese edited by Neal Robbins

and published by Yale University Press. University of Illinois adopted "The Amateur Cameraman" in course materials for studying Chinese. Dr. Chang is also regarded as the father of science fiction in Taiwan. Some of Dr. Chang's SciFi short stories have been translated into English, such as "City of the Bronze Statue" , "Love Bridge" , and "Returning" . His SciFi novel, The City Trilogy, was published by Columbia University Press in May 2003.

# Web Services Reliability Patterns

Ingrid Buckley<sup>1</sup>, Eduardo B. Fernandez<sup>1</sup>, Gustavo Rossi<sup>2</sup> and Masoud Sadjadi<sup>3</sup>

<sup>1</sup> Florida Atlantic University, Boca Raton, Fl, 33431, USA, {ibuckley,ed}@cse.fau.edu

<sup>2</sup> Universidad Nacional de La Plata, Argentina, La Plata, gustavo@lifa.info.unlp.edu.ar

<sup>3</sup> Florida International University, Miami, Fl, 33199, USA, sadjadi@cs.fiu.edu

## Abstract

*Due to the widespread use of web services by enterprises, the need to ensure their reliability has become crucial. There are several standards that intend to govern how web services are designed and implemented, including protocols to which they must adhere. These standards include the WS-Reliability and WS-Reliable Messaging standards that define rules for reliable messaging. We present here patterns for these standards which define how to achieve reliable messaging between entities. We compare their features and use.*

**Keywords:** Web Services, Reliability, Patterns.

## 1. Introduction

Web services have become the most popular means used by enterprises to offer services to their customers and to interoperate with business partners. These services are accessed through messages. Since messaging is crucial to the enterprise in terms of the services and transactions that are exchanged between businesses and customers, it has become essential to ensure reliable messaging. Reliable messaging, as used in this context, is the act of sending a message without duplication, ensuring guaranteed delivery as well as message ordering and message state disposition [Oas04, Oas07]. The implications of a failure in this respect can have a damaging impact on businesses that rely on the availability and reliability of the services offered to customers.

In general, standards defined by committees are rather complex and their descriptions are given in uniform detail, which together with their length, make understanding of the standards rather difficult. In particular, web services standards are expressed using XML, are relatively complex and lengthy, between 57 and 120 pages. By expressing these standards as patterns, including precise UML models, we attempt to make them more understandable and easier to compare to other standards with similar objectives.

The WS-Reliability and WS-Reliable Messaging Standards are defined by OASIS and the former has borrowed from the ebXML Message Service Specification

2.0 technology. WS-Reliability is a SOAP-based (SOAP 1.1 and SOAP 1.2 Part 1) specification that fulfills reliable messaging requirements critical to some applications of Web Services [W3c07]. The WS-Reliability standard utilizes quality of service (QOS) contracts, and uses conditions attached to the invocation of a set of operations; namely deliver, submit, respond and notify [Oas04]. To perform reliable delivery it uses the concept of Reliable Message Processor (RMP). The WS-Reliable Messaging standard provides guaranteed delivery, message ordering and duplicate elimination [Oas07]. To support interoperable web services, a SOAP binding is defined within this specification. However the protocol depends upon other web services specifications for identification of service endpoint addresses and policies [Oas07]. It is also possible to consider reliability at the higher levels, for example [Dob06].

The rest of this paper is organized as follows. Section 2 presents the WS-Reliability pattern, and section 3 presents the WS-Reliable Messaging pattern. Section 4 compares these two standards. We end with some conclusions in Section 5. We show only parts of the patterns for lack of space. A more complete report is available from the first author [Buc08].

## 2. WS-Reliability

### 2.1 Intent

WS-Reliability ensures that a notification is always sent in response to a failure, it also provides guaranteed message delivery, message ordering, and duplicate elimination whenever messages are sent from one entity to another.

### 2.2 Context

Institutions, business-to-business (B2B) applications, and critical infrastructure systems that need to send and receive messages in real-time.

**2.3 Problem** Some applications need reliable messaging in order to fulfill their business operations effectively and successfully. Many people communicate via the internet, thus creating heavy network traffic. Many companies offer services to consumers across the internet, which gives rise to bandwidth and availability problems. Enterprises are concerned about how to achieve reliable messaging given

some of the factors mentioned previously; more specifically, to ensure that messages are delivered with acknowledgment of receipt, in the order sent, and without duplication. How do we ensure that messages that are sent are delivered, acknowledged, sent in order, and without duplication? The solution to this problem is affected by the following *forces*:

- Dissimilar internet connection speed used by both sides (receiving and sending parties) can affect how quickly messages are sent and received.
- Network traffic affects the time it takes a message to reach a recipient; this may increase the delay time for the messages and may change their order.
- The receiving or sending party may become unavailable and some or all messages may not get sent or received.
- Unordered and delayed messages can lead to problems for online transactions especially in banking systems and critical infrastructures.
- The response time to messages contributes to delay; when messages get lost or arrive to a recipient unordered, the recipient may take more time to respond, thus increasing the delay time.

## 2.4 Solution

Use a protocol with acknowledgement of delivery or failure, message ordering, and duplicate message elimination. This is achieved by first having an enforceable contract between the sending and receiving parties, and the use of sending and receiving reliable message processors (RMPs) that send, deliver order and eliminate duplicate messages.

The WS-Reliability standard utilizes four primary conceptual units as illustrated in Figure1. The Producer creates and submits messages to the Sending RMP. The Consumer receives messages delivered by the Receiving RMP and sends an acknowledgement. The Sending RMP submits messages to and receives acknowledgements from the Receiving RMP. The Receiving RMP is responsible for delivering messages to the consumer and receiving and sending notification from the consumer to the Sending RMP. A QoS Contract binds the agreement made between the consumer and producer. A protocol contract binds the Sending and Receiving RMP.

### Structure

A **contract** (Figure 2) defines the quality of service expected between the sending and receiving RMP as well as the terms of the relationship between the **Producer** and the **Consumer**. The contract includes a specification of the expected quality of service (**QoS**), which determines the quality of messaging service to the communicating parties, and the **Features** which define the operations and rules which are expected. The **Reliable Messaging Processor (RMP)** [Oas04] handle messages that are sent between a producer and a consumer and perform

messaging as outlined in the contract in the form of requirements such as guaranteed delivery, duplicate message elimination, and message ordering. The implementation of the RMP is not specified by the standard, and can be implemented in many different ways (see implementation).

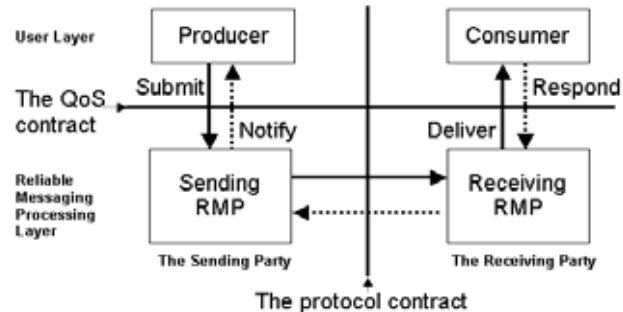


Figure 1: Structure and dataflow of the components involved in the WS-Reliability Standard [Oas04]

Processes may have two **ProcessRoles**, the **Producer** role creates messages and sends them to the **Sending RMP**. The **Consumer** role consumes messages that have been processed by the **Receiving RMP**. A **Message** can be a **Group Message** or an **Individual Message** with varying attributes depending on the type of message. The **SOAP MessageExchangePattern (MEPs)** defines different modes of response which can be sent from the Consumer to the Receiving RMP in response to a previously received message. The SOAP MEPs used is defined in SOAP 1.2 [W3c07].

### Dynamics

Use cases Send a message and Establish an agreement are not shown for lack of space.

## 2.5 Consequences

The WS-Reliability pattern presents the following advantages:

- Messages sent between end points can be controlled by means of the RMP that ensures delivery with acknowledgment, ordering, and duplicate elimination of messages within the limits imposed by the network.
- Enterprises are able to obtain a higher degree of reliability for network communication because the sender and receiver confirm reception by an acknowledgment each time they communicate via a message.
- Quality of service defined by contracts can be maintained between businesses thus increasing reliability and supporting the accountability of business partners. Policies can be attached to the contracts that govern the modus operandi agreed by all communicating parties.

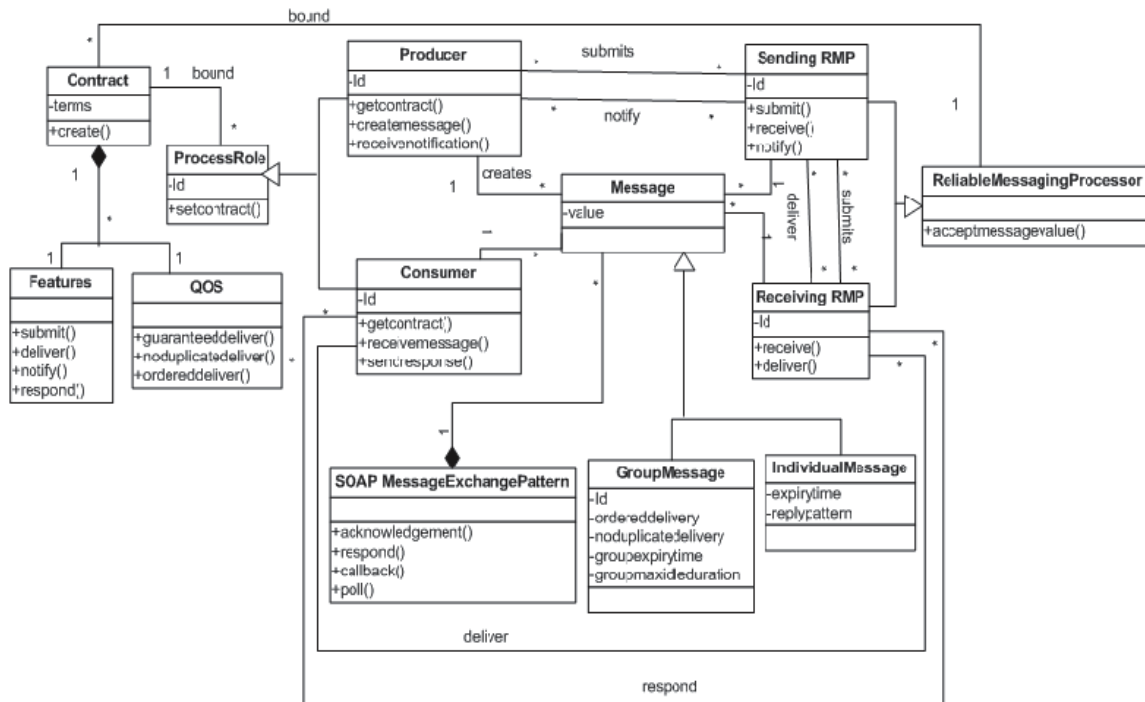


Figure 2: Class Diagram for the WS-Reliability pattern

The RMP sends an acknowledgment if the consumer becomes unavailable during the transmission of a reliable message.

The pattern also has some possible liabilities:

- The message and its response are passed between several components and not directly to the recipient or producer of the message. This process increases the time it takes the message to be delivered to the recipient and the time it takes to send the corresponding notification back to the producer of the message.
- The WS-Reliability pattern increases the complexity in the system.

### 3. WS-Reliable Messaging

#### 3.1 Intent

WS-Reliable Messaging ensures guaranteed receipt in response to each message sent; it also provides, message state disposition, ordered delivery, and duplicate elimination whenever messages are sent between endpoints.

#### 3.2 Context

Institutions, B2B applications, and critical infrastructure systems that need to send and receive messages in real-time.

#### 3.3 Problem

Many errors can interrupt communication, messages can get lost, duplicated, or reordered; the host system may experience failures and lose volatile state and messages may also experience state loss during transmission.

Some applications need to have reliable messaging in order to fulfill their business operations effectively and successfully; therefore, lost, unordered and duplicate messages can have a negative affect on successful business operations. How do we ensure ordered delivery, guaranteed receipt, duplicate elimination and state disposition of messages? The solution to this problem is affected by the following forces:

- The receiving or sending host may become unavailable and some or all messages may not get sent or received.
- Messages may get lost during transmission.
- Unordered and delayed messages can lead to problems for online transactions especially in banking systems and critical infrastructures.
- The response time to messages contributes to delay in sending a receipt; when messages get lost or arrive to a recipient out of order, it may take more time to respond, thus increasing the response time.
- Dissimilar internet connection speed used by both sides (receiving and sending parties) can

affect how quickly messages are sent and received.

- Network traffic affects the time it takes a message to reach a recipient; this may increase the delay time for the messages and may change their order.

### 3.4 Solution

Use a protocol that performs guaranteed receipt, ordered delivery, state disposition, and duplicate elimination of messages. This is achieved by first having an agreement which includes a policy exchange, endpoint resolution and establishment of trust between end points.

The WS-Reliable Messaging standard utilizes four primary conceptual units as illustrated in Figure 3.

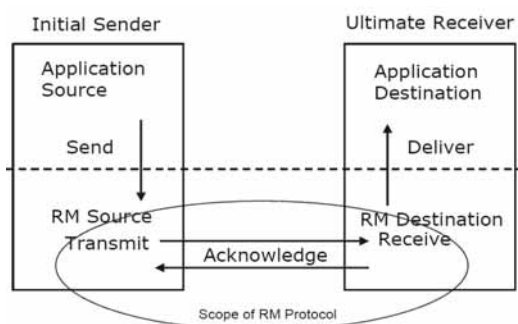


Figure 3: Structure and dataflow of the components involved in the WS-Reliable Messaging Standard [Oas07]

The application source creates and sends messages to the RM Source. The RM Source transmits messages to the RM Destination. The RM Destination receives messages transmitted from the RM source and sends a corresponding receipt of acknowledgement; the message is then delivered to the application destination/receiver.

#### Structure

An **Agreement** enforces policy exchange, end point resolution, and trust establishment between the **Application Source** and the **Application Destination**. The **Application Source** creates and sends messages to the RM source (Figure 4). A **Message** consists of content and information about where it is supposed to be delivered. The RM Source transforms a message into a **Reliable Message** by adding new properties to the message. A **Sequence** (created by the RM destination at the request of the RM Source) acts like an envelope in which a Reliable message is placed before it is transmitted. The **RM Source** accepts messages and acknowledgements from the Application Source and RM Destination respectively, and transmits reliable messages to the RM destination. The **RM Destination** receives messages sent from the RM Source, sends a

corresponding acknowledgement of receipt to the RM Source, and delivers the reliable message to the destination application. The **Application Destination** receives reliable messages from the RM Destination.

#### Dynamics

Use cases Send a message and Establish an agreement describe dynamic aspects but are not shown for lack of space.

### 3.5 Consequences

The WS- Reliable Messaging pattern presents the following advantages:

- Enterprises are able to obtain a higher degree of reliability for network communication because endpoints create and terminate message sequences. In addition a receipt of acknowledgement is sent every time a message is sent and retransmission of messages is done for messages that were not received.
- Quality of service defined by agreements can be maintained between businesses, thus increasing reliability and supporting the accountability of business partners.
- The WS-Policy standard is used to govern policies that can be attached to the agreements that govern the operations agreed to by communicating endpoints, therefore leveraging the use of other web service standards.
- WS-Addressing is utilized to achieve endpoint referencing. This specifies the endpoint reference to where the receipt of acknowledgement is to be sent in response to a message. In this way messages cannot be intercepted easily because the destination is known prior to their transmission.
- Terminate message sequence requests are sent to the RM destination to notify when no more messages will be sent using a given sequence. Therefore the system resources attached to a sequence can be freed and used to conduct other operations.

The pattern also has some possible liabilities:

- Introduces a high time overhead with the retransmission of messages and acknowledgements. The RM Source will retransmit messages for which no receipt of acknowledgments was received. This could result in high volume requests flooding the RM Destination depending on the retransmission and back-off interval set.
- There is a high demand on the resources used to track the state of each message transmitted as required by the RM Source.



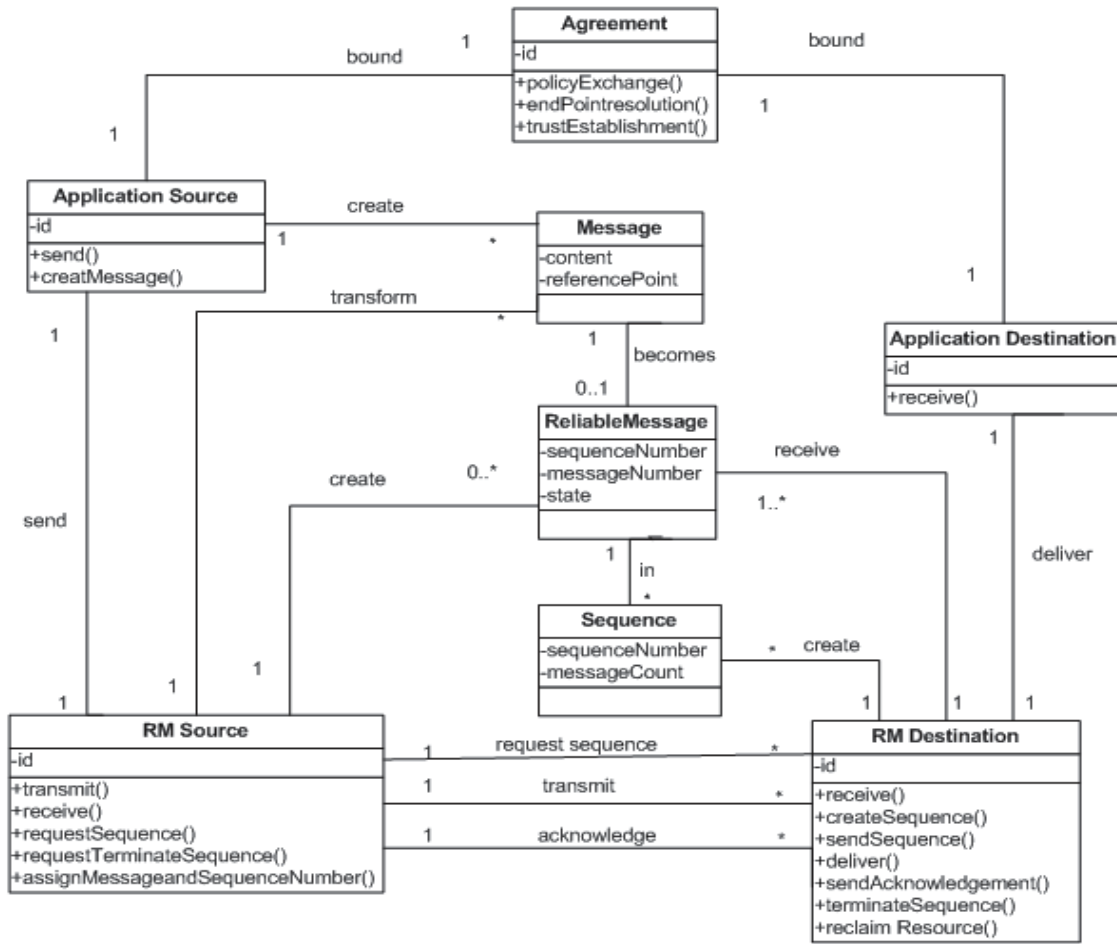


Figure 4: Class Diagram for the WS-Reliable Messaging pattern.

#### 4. Comparing the WS-Reliability and WS-Reliable Messaging Patterns

WS-Reliability and WS-Reliable Messaging specifications offer the same basic service, which is sending messages in a reliable manner. However, the two protocols utilize different means of performing this service. WS-Reliability has a binding to HTTP whereas WS-Reliable Messaging is transport independent allowing it to be implemented using different network technologies. In order to support interoperable web services, a SOAP binding is defined within both patterns. The specifications mandate that an agreement be made before communication can be done between endpoints. However the WS-Reliable Messaging explicitly states that endpoint referencing, establishment of trust and policy exchange are to be included in the agreement. Endpoint reference explicitly states the address where a reliable message should be sent. Establishment of trust is achieved with an enforced agreement and policy exchange facilitates the updating of quality of service terms and conditions. WS-Reliability does not explicitly dictate the terms of the contract.

WS-Reliability engages the producer and consumer of a message in the entire cycle of sending a reliable message; due to the fact that WS-Reliability ensures that a

guaranteed acknowledgement be sent to the producer of a reliable message. The producer specifies the mode of response that is required from the consumer and waits until an acknowledgement is received, this acknowledgement ends the cycle. In contrast, WS-Reliable Messaging ensures guaranteed receipt; the RM Source and Destination components control the execution of a reliable message between each other. Once the initial message is obtained, a guaranteed receipt is sent between these two components, not directly to the initial sender. In other words, WS-Reliable Messaging does not require that the sender listens for a guaranteed receipt, this is dealt with by the RM Source.

Additionally WS-Reliable Messaging must use a sequence to transmit all messages (individual and series), while in WS-Reliability the Sending RMP and Receiving RMP send messages either individually or in groups. Another contrast between the two specifications is that WS-Reliable Messaging mandates that all sequences be ended when no further messages will be sent using that sequence. This allows resources that are attached to each sequence to be reclaimed. WS-Reliability uses a GroupExpiryTime to terminate group messages and an ExpiryTime to terminate an individual message.

Another difference between the two specifications is that WS-Reliability uses SOAP message exchange patterns, which specify the mode of response to be used by the recipient of a reliable message. The message exchange patterns used are poll, respond and callback. However, WS-Reliable Messaging does not explicitly ask for a particular response mode from the recipient of a reliable message. In fact WS-Reliable Messaging does not require a response from the recipient of a reliable message, because the RM Destination sends a receipt of acknowledgement to the RM Source directly. Additionally WS-Reliable Messaging allows a receipt of acknowledgment to be sent with or without using the SOAP body.

In summary, WS-Reliability will only send an acknowledgement when a reliable message is delivered to the recipient; this supports real time communication using messaging. However WS-Reliable Messaging sends a receipt of acknowledgment once the RM destination receives a reliable message, which can be done before, after or simultaneously to delivering the reliable message to its destination. In the case of group messages WS-Reliable Messaging can hold messages at the RM Destination until all messages are received and send them all at once to the recipient. Therefore the concept of guaranteed acknowledgment and guaranteed receipt is different between the two specifications.

WS-Reliable Messaging is dependent on WS-Policy and WS-Addressing for policy and identification of endpoints respectively. WS-Reliability and WS-Reliable Messaging do not explicitly state how to achieve ordering and duplicate elimination of messages. Instead, they state that both features can be implemented in different ways. WS-Reliable Messaging includes message state disposition, the RM Source tracks each reliable message until a receipt is received from the RM Destination. WS-Reliability uses a message number to keep track of each message sent.

WS-Reliability requires a contract while WS-Reliable Messaging requires an agreement before communication can begin between endpoints. The patterns are similar in this regard however, the agreement includes establishment of trust, policy exchange and endpoint resolution. The structure of the key components in the WS-Reliability and WS-Reliable Message patterns are similar, see Figure 2 and 4. However the WS-Reliability pattern uses the Sending and Receiving RMP to provide acknowledgement, ordering, duplicate elimination, and guaranteed delivery of messages, while, the WS-Reliable Messaging pattern uses the RM Source and RM Destination to provide similar functions.

## 5. Conclusions

WS-Reliability and WS-Reliable Messaging are two standards intended to specify the reliable delivery of

message between web services. We have provided patterns for these standards. The original standards are verbose and complex; we hope to have clarified their structure and behavior. Since both standards apply to the same problem we provided a comparison of their features.

Future work will include the development of further patterns so as to provide the designer with a catalog of patterns that can be used when developing web-services-based systems.

## Acknowledgements

This work was supported in part by IBM, the National Science Foundation (grant OISE-0730065), and DoD's DISA through Pragmatics, Inc.

## References

- [Buc08] I. Buckley and E.B Fernandez, "Web Services Reliability Patterns", Department of Computer Science and Engineering, Florida Atlantic University, 2007.
- [Dob06] G. Dobson, "Using WS-BPEL to implement software fault-tolerance for Web Services", Proc. 32<sup>nd</sup> EUROMICRO Conference on software, 2006.
- [Gam94] E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Boston, Mass., 1994.
- [Oas04] OASIS, "Web Services Reliable Messaging TC WS-Reliability 1.1", [http://docs.oasis-open.org/wsrn/ws-reliability/v1.1/wsrn-ws\\_reliability-1.1-spec-os.pdf](http://docs.oasis-open.org/wsrn/ws-reliability/v1.1/wsrn-ws_reliability-1.1-spec-os.pdf), 2004.
- [Oas07] OASIS, "Web Services Reliable Messaging (WS-Reliable Messaging) Version1.1", <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.pdf>, 2007.
- [W3c07] W3C, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)" <http://www.w3.org/TR/soap12-part1/>, 2007

# Consistency in Self-Reconfiguration of Self-Healing Systems

Michael E. Shin  
*Dept. of Computer Science*  
*Texas Tech University*  
*Lubbock, TX 79409-3104*  
*Michael.Shin@ttu.edu*

Kiran Gopala Reddy Sunanda  
*Dept. of Computer Science*  
*Texas Tech University*  
*Lubbock, TX 79409-3104*  
*kiran.gs@ttu.edu*

## Abstract

This paper describes the consistency in the self-reconfiguration of component-based self-healing systems against anomalous objects in the components. A self-healing component is structured into different types of objects. When a self-healing component meets some faults, it self-reconfigures objects by isolating anomalous objects from healthy objects within the component. The self-reconfiguration may cause a component to be inconsistent due to anomalous objects. This paper describes an approach to establishing consistency at runtime in the self-reconfiguration of component-based self-healing systems. Self-healing components remain consistently in the self-reconfiguration by rolling back an inconsistent state to a consistent state. For this, objects in self-healing components are designed to support the consistency of self-reconfiguration.

## 1. Introduction

The critical software systems need to have the capability that makes the systems reliable. One of the approaches to making the systems more reliable is the self-healing mechanism [Dashofy02, Garlan03, IBM03, Koopman03, Shin05], which involves detection, reconfiguration, and repair of faults or unanticipated events that may lead the systems to a failure. A system having the self-healing mechanism detects faulty objects autonomously, self-reconfiguring the system against the faults detected, and repairing the faults at runtime so that the system continues to provide its services.

The self-reconfiguration of a self-healing component against a faulty object may cause the system to be inconsistent due to incompletely processing the service requests within the component. When an object in a self-healing component does not finish processing a service request from other objects, the self-healing component should be rolled back to a consistent state. Without this rollback step, the unfinished service request can make the component fall into an inconsistent state.

Several approaches have been suggested to resolve the consistency in the reconfiguration of systems. These approaches [Kramer90, Feiler98, Almeida01, Palma02,

Gomaa04, Rasche05] mainly focus on the consistency in the reconfiguration for evolution or change of systems. However, less attention has been paid to the consistency of self-reconfiguration in the self-healing systems.

This paper describes an approach to maintaining self-healing components consistently in the self-reconfiguration for self-healing systems when some objects in the components have faults or come across unanticipated events. In this paper, a consistent self-reconfiguration approach is developed based on the reconfiguration mechanism [Shin06] for component-based self-healing systems. The self-reconfiguration approach suggested in [Shin06] does not deal with consistency of self-reconfiguration.

## 2. Self-Reconfiguration of Self-healing Components

A self-healing component is able to autonomously detect, reconfigures, and repair anomalies on itself as well as provides functional services to other components. A self-healing component [Shin05, Shin06] is structured into both the service layer and healing layer in which the service layer is separated from the healing layer. Fig. 1 depicts the self-healing component architecture. The service layer of a self-healing component provides functional services to other components, notifying the status of messages passing between objects in that layer to the healing layer. The healing layer detects anomalies in the service layer using the notification messages from the service layer, and reconfigures and repairs the detected anomalies.

The service layer of each self-healing component is composed of objects such as tasks (active or concurrent objects), passive objects accessed by tasks, and connectors between tasks. A task depicted using a thick outline for the object box in Fig. 1 has its own thread of control, initiating actions that affect other tasks and passive objects [Gomaa00]. Unlike a task, a passive object (e.g., an entity object) has no thread of control; thus it cannot initiate any task. Since a passive object does not have its own thread, it performs its operations using the thread of the task that invoked the object. A passive object invoked by a task can call other passive

objects using the thread of the task as well. Tasks in a component can communicate with each other through connectors. On behalf of tasks that have threads, a connector between tasks synchronizes the message communication between the tasks. Passive objects accessed by tasks and connectors between tasks, which do not have their threads, are depicted using thin outlines for the object boxes in Fig. 1.

The healing layer of each self-healing component (Fig. 1) is structured into *Component Monitor*, *Reconfiguration Manager*, *Repair Manager*, and *Self-Healing Controller*, which are responsible for detection, reconfiguration, and repair of anomalous objects in the service layer.

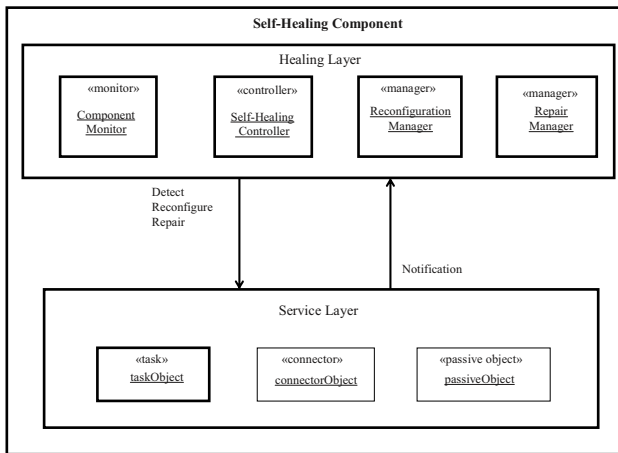


Fig. 1 Self-Healing Component Architecture

A self-healing system is reconfigured at the two levels - component level and connector level between components. The reconfiguration of anomalous objects in a component is performed by the Reconfiguration Manager in the healing layer of the component (Fig. 1). The Reconfiguration Manager in the healing layer of a component generates a reconfiguration plan against anomalous objects on the basis of the current configuration of the component. Using the plan, the Reconfiguration Manager blocks objects associated with the anomalous objects and, if needed, notifies the anomalies of objects to neighboring components affected from the reconfiguration of the paralyzed component. In response to this notification, the Reconfiguration Managers of the neighboring components also generate their reconfiguration plans and undertake the plans to reduce impact from the paralyzed component.

An anomalous task in the service layer of a self-healing component is self-reconfigured by blocking the incoming connectors to the task and outgoing connectors from it. The Reconfiguration Manager in the healing layer sends a message to the incoming connectors so that the task cannot read a service request message in the queues or buffers in the incoming connectors. Similarly,

the Reconfiguration Manager sends a message to the outgoing connectors so that the anomalous task cannot add messages to the queues or buffers in the outgoing connectors. The messages from the Reconfiguration Manager makes the connectors isolate the anomalous task.

An anomalous passive object accessed by tasks is self-reconfigured by preventing the tasks from invoking operations of the passive object. Anomalies in a passive object accessed by tasks are detected when a task invokes an operation of the passive object. After the passive object notifies the invocation of an operation to the Component Monitor (Fig. 1), the operation may not be performed successfully so that the passive object cannot send the Component Monitor the next notification message showing the complete finish of the operation.

For isolating the anomalous passive object, the task invoked the anomalous operation of the passive object needs to be blocked and the thread of the task should be interrupted to terminate immediately. The Reconfiguration Manager creates a new thread for the task, but the thread cannot access the anomalous object because the thread checks the status of the passive object before invoking an operation provided by the passive object. The task blocked cannot call the operation again until the anomalous operation or the entire passive object is self-healed.

An anomalous connector between tasks is self-reconfigured by blocking the sender task and receiver task accessing the connector. The sender task is blocked immediately when it invokes the anomalous send() operation of a connector to send a message to the receiver task. The receiver task may not be blocked immediately depending on the type of connectors such as a message queue connector (queue size is n), message buffer connector (buffer size is 1), and message buffer and response connector (both message buffer and response buffer sizes are one respectively). In the case of a message queue connector, the receiver task should be blocked after processing all the messages in the queue of the connector. This is to prevent messages already stored in the queue from being lost. In the other cases, the receiver task needs to be blocked immediately because there is no message in buffers. On the other hand, a sender task and a receiver task should be blocked immediately when a receiver task calls an anomalous operation in a connector. This is because new messages delivered by a sender task cannot be read by a receiver task any more.

### 3. Consistent Self-Reconfiguration

The self-healing components can be consistently self-reconfigured using the checkpoint and rollback pattern

[Hanmer07]. A checkpoint is either a rollback point or a synchronization point. Objects in a self-healing component store a service request message at a checkpoint until the message is completely processed. If a rollback occurs, the message still exists in objects and can be restored again. On the other hand, some operations in objects are confirmed to commit the operations finally at a checkpoint as a synchronization point if a message is completely processed. Otherwise, the operations are aborted.

A connector between tasks in the service layer of self-healing component has checkpoints for consistent self-reconfiguration of self-healing components. A task sends a message to another via a connector in which the message is saved temporarily at a variable in a connector between the tasks. When a task has processed a message completely, it notifies connectors. With the confirmation message from a task, the connector saves the message stored temporarily in a variable to a queue or buffer permanently so that the message is delivered to the receiving task. Similarly, when a task receives a message from a connector, the message is saved temporarily at a variable until the message is completely processed.

A passive object accessed by tasks in the service layer of self-healing component provides a checkpoint to maintain consistency in self-reconfiguration of self-healing components. When a write() operation in a passive object is invoked by a task, the write() operation is processed and saves the processed result temporarily at a variable. When the passive object receives a confirmation message from a task, it processes the write() operation permanently.

Figure 2 depicts a scenario that a task processes a service request message using the UML collaboration diagram [Booch05, Rumbaugh04]. When the Task1 (Figure 2) reads a Message1 from a queue of the Connector1, the message is temporarily saved in a variable in the Connector1. The Task1 may need to read the Data1 and write the Data2 in the PassiveObject so as to process a service request Message1. In this case, the write() operation in the PassiveObject is performed and the result is stored temporarily in a variable. Then the Task1 may send a Message2 to another task through the Connector2 in which the message is temporarily stored in a variable. When the Task1 finishes processing the Message1 completely, it sends confirmation messages to the Connector1, PassiveObject, and Connector2. With these confirmation messages, the Message1 stored temporarily in a variable of the Connector1 is cleared permanently; the write() operation of the PassiveObject is committed permanently; and the Message2 temporarily stored in the Connector2 is added to the queue permanently.

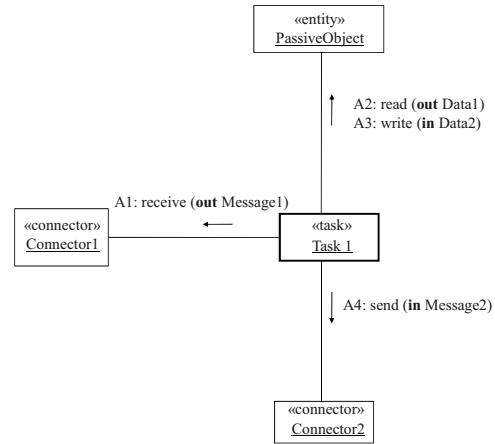


Fig. 2 Scenario of handling a service request message

When a fault is found in an object of a self-healing component, the state of objects that participate in processing a service request message not completed due to the fault is rolled back to the state just before the message is processed. The rollback of state is accomplished by clearing or returning the message stored temporarily in a variable to a queue in a connector. This is because loss of the message may cause inconsistency of a component in the self-reconfiguration. In case where a passive object accessed by tasks is involved in the processing of the message, it clears the message stored in a temporary variable for a write() operation.

The Component Monitor (Fig. 1) detects some faults in the objects associated with processing a service request message. The Component Monitor notifies the detection of a fault in an object to the Reconfiguration Manager in the healing layer (Fig. 1), which reconfigures the objects consistently. For this, the Reconfiguration Manager in the healing layer communicates with connectors, passive objects and tasks in the service layer that are related to the fault.

### 3.1 Consistent self-reconfiguration of anomalous connector between tasks

A connector between tasks provides send() and receive() operations so that the message sender task sends a message to the message receiver task synchronously or asynchronously. The send() operation is invoked by the message sender task to store a message in a queue or buffer in a connector. The receive() operation is called by the message receiver task to read a message from a queue or buffer in the connector.

In case where the send() operation in a connector is anomalous, the Reconfiguration Manager clears the message that might be stored temporarily in a variable of the connector, making other objects participating in the processing of the message roll back to a consistent state.

Consider the scenario where the send() operation in the Connector2 is anomalous in Fig. 2 (that is, A4 will not finish successfully) after the message sequence A1 through A3 has completed. Although the send() operation is anomalous, the Message2 might be stored temporarily in a variable in the Connector2. The message should be cleared by the Reconfiguration Manager. The work done by the Task1 also needs to be undone by the Reconfiguration Manager so that the objects associated with the Task1 should roll back to the consistent state. In Fig. 2, the message not successfully processed by the Task1 is back to the queue in the Connector1. The PassiveObject clears the results stored temporarily after performing the write() operation.

When the receive() operation in a connector is anomalous, a message might be stored to a temporary variable during the partial processing of the anomalous receive() operation. The message should be restored by the Reconfiguration Manager to the queue or buffer so that it is processed again after the anomaly is repaired. For instance, the Message1 may need to be restored to the queue if the receive() operation provided by the Connector1 (Fig. 2) is anomalous when it is called by the Task1. Similar to the case of anomalous the send() operation, other objects, if any, participating in the processing of the message should be rolled back by the Reconfiguration Manager to the consistent state.

### 3.2 Consistent Self-reconfiguration of Anomalous Passive Object

The Reconfiguration Manager needs to reconfigure objects participating in the processing of an unfinished service request message due to anomaly of either a read() or write() operation in a passive object accessed by tasks. A write() operation in a passive object accessed by tasks may meet some fault just after it stores the partial result temporarily to some variable. This partial result should be cleared by the Reconfiguration Manager. In addition, the work done in related objects just before the anomaly should be rolled back to the consistent state by the Reconfiguration Manager.

Suppose the write() operation of PassiveObject is anomalous in the mid of processing the Message1 by the Task1 in Fig. 2. The write() operation will not finish successfully. The Reconfiguration Manager makes the PassiveObject clear the partial result that might be stored in a temporary variable, having the Connector1 restore the Message1 to the queue or buffer. Similarly, in case of anomalous read(), the Reconfiguration Manager makes the Connector1 restore the Message1 to the queue or buffer.

### 3.3 Consistent Self-reconfiguration of Anomalous Task

When a task comes to be anomalous, the Reconfiguration Manager makes the state of related objects roll back to a consistent state. Suppose the Task1 in Figure 2 becomes anomalous when the Task1 is processing just after writing the Data2 in the PassiveObject. The Component Monitor detects some fault in the Task1 and the Reconfiguration Manager starts the consistent reconfiguration. The Connector1 and PassiveObject will receive the messages about the anomaly of Task1 from the Reconfiguration Manager. The messages indicate that the Connector1 and PassiveObject need to undo all the changes and activities done until the detection of anomaly. The receive() operation performed in the Connector1 by the Task1 must be undone by moving the Message1 stored temporarily in a variable to the queue in the Connector1. The Message1 stored in the temporary variable must also be cleared in the Connector1. The write() operation carried out by the Task1 in the PassiveObject must be undone by clearing the processed result stored temporarily in a variable.

## 4. Design of connectors and passive objects

Connectors and passive objects accessed by tasks are designed to support the consistent self-reconfiguration approach described in section 3.

### 4.1 Consistent Message Queue Connector

The message queue connector for consistent self-reconfiguration (MessageQueueSHWithConsistency class in Figure 3) is designed by specializing it from the message queue connector [Shin06] that does not support the consistency in the self-reconfiguration mechanism (MessageQueueSH class in Figure 3). The message queue connector for consistent self-reconfiguration provides additional attributes and operations to maintain consistency of the connector.

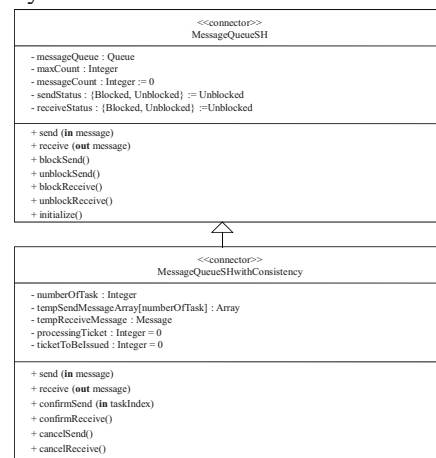


Fig. 3 Message Queue Connector with Consistency

The attributes used in the design of message queue connector for consistent self-reconfiguration (MessageQueueConnectorWithConsistency in Figure 3) are described as below:

- The numberOfTask is a variable used to keep track of the number of tasks storing messages in the message queue.
- The tempSendMessageArray is an array that temporarily holds messages being stored in the queue.
- The tempReceiveMessage is a variable that temporarily holds a message until it is processed successfully. Otherwise, the value of this variable is restored to the queue.
- The processingTicket and ticketToBeIssued are variables used to handle multiple tasks that send messages to a receiver task through a shared connector.

The MessageQueueConnectorWithConsistency class in Figure 3 has the following operations additionally:

- The send() operation is modified to store a message in the tempSendMessageArray temporarily.
- The receive() operation is redefined to store a message in a tempReceiveMessage variable when the receiver task read a message from the queue.
- The confirmSend() operation is used to add a message stored temporarily in the tempSendMessageArray to the messageQueue permanently.
- The confirmReceive() operation clears the value stored in the tempReceiveMessage variable.
- The cancelSend() operation is called by the Reconfiguration Manager to clear a message in the tempSendMessageArray when the system fails to process a message.
- The cancelReceive() operation is called by the Reconfiguration Manager to restore a message stored in the tempReceiveMessage to the queue when the system fails to process a message.

#### 4.2 Passive Object with Consistent Self-reconfiguration

It is assumed that passive objects accessed by tasks have read and write operations. The write operation performed in the passive objects must be tracked to maintain consistency of self-reconfiguration when the system fails to process a message. The passive object will have the following special operations confirmWrite() and cancelWrite() to establish the consistency. Figure 4 depicts a passive object (class) for consistent self-reconfiguration (PassiveObjectSHwithConsistency class)

that is specialized from a passive object (class) for non-consistent self-reconfiguration.

- The write() operation stores the result to the tempWriteData variable temporarily after processing the write() operation in the PassiveObjectSH.
- The confirmWrite() operation writes the value stored temporarily in the tempWriteData to the writeData variable permanently.
- The cancelWrite() operation cancels the write() operation by clearing the value stored in the tempWriteData variable.

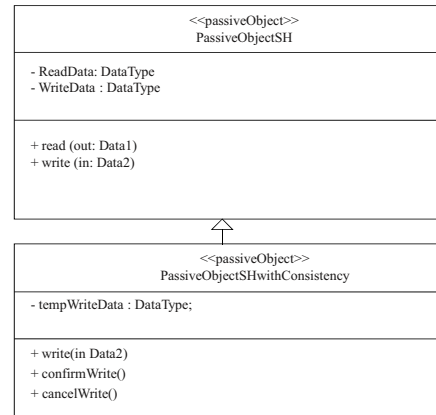


Fig. 4 PassiveObjectSHwithConsistency Class Diagram

### 5. Discussion

The consistent self-reconfiguration approach described in this paper has been applied to the self-healing elevator system with multiple elevators, which is structured with the Elevator Control, Scheduler, and Floor components. The components in the elevator system communicate with each other through consistent message queue connectors designed in section 4.1. The consistent self-reconfiguration approach was implemented on the self-reconfigurable elevator system that was developed in our previous research without considering the consistency of self-reconfiguration. Our approach was tested by inserting faults to each object type such as a connector, task, or passive object.

Although our approach provides the consistent self-reconfiguration, it may have some weaknesses. Addition of a message to and retrieval of a message from a message queue connector may be delayed until a task handles the message successfully. In a passive object supporting the consistency of self-reconfiguration, some information may not be updated on time. The execution of a write() operation may be delayed until a task finishes its activity successfully. In addition, write() operations in all passive objects should be modified to support the consistent self-reconfiguration. For each write() operation

in a passive object, a pair of operations, `confirmWrite()` and `cancelWrite()`, should be defined additionally.

## 6. Conclusions

This paper has described an approach to maintaining self-healing components consistently in the self-reconfiguration for self-healing component-based systems against some faults in objects constituting the components. Self-healing components remains consistently in the self-reconfiguration by rolling back an inconsistent state to a consistent state. For this, objects in components are designed to have additional functionality to support the consistency in the self-reconfiguration. The consistent self-reconfiguration approach has been applied to the elevator system.

This paper has further research. The approach proposed in this paper should be validated more by applying to other application systems. The consistent self-reconfiguration approach can be revised based on the experience from other applications. Another direction for further research is to develop different type of connectors supporting consistent self-reconfiguration. This paper focused on asynchronous connector, but not included synchronous connectors such as a message buffer connector.

## References

- [Almeida01] Almeida, J. P. A., Wegdam M., P. F. Luís, and M. V. Sinceren, "An approach to dynamic reconfiguration of distributed systems based on object-middleware," Proceedings of 19th Brazilian Symposium on Computer Networks, 2001.
- [Booch05] Booch, G., J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language User Guide", Second Edition, Addison Wesley, Reading MA, 2005.
- [Dashofy02] Dashofy, E. M., A. V. D. Hoek, and R. N. Taylor, "Towards Architecture-based Self-Healing Systems," Workshop on Self-healing systems, Proceedings of the first workshop on Self-healing systems, Charleston, SC, November 18-19, 2002.
- [Feiler98] Feiler, P. and J. Li, "Consistency in Dynamic Reconfiguration", Fourth International Conference on Configurable Distributed Systems, Annapolis, MA, USA, May 1998.
- [Garlan03] Garlan, D. S. Cheng, and B. Schmerl, "Increasing System Dependability through Architecture-based Self-repair," in *Architecting Dependable Systems*. R. de Lemos, C. Gacek, A. Romanovsky (Eds), Springer-Verlag, 2003.
- [Gomaa00] Gomaa, H., "Designing Concurrent, Distributed, and Real-Time Applications with UML", Addison-Wesley, 2000.
- [Gomaa04] Gomaa, H. and M. Hussein, "Software Reconfiguration Patterns for Dynamic Evolution of Software Architectures", Proc. Fourth Working IEEE/IFIP Conference on Software Architecture, Oslo, Norway, June, 2004.
- [Hanmer07] Hanmer, R., "Patterns for Fault Tolerant Software," Wiley, 2007.
- [IBM03] IBM, "An architectural blueprint for autonomic computing," IBM and autonomic computing, April 2003.
- [Koopman03] Koopman, P., "Elements of the Self-Healing System Problem Space," Workshop on Software Architectures for Dependable Systems (WADS2003), ICSE'03 International Conference on Software Engineering, Portland, Oregon, May 3-11, 2003.
- [Kramer90] Kramer, J. and J. Magee, "The Evolving Philosophers Problem: Dynamic Change Management," IEEE TSE 16, 11, November 1990.
- [Palma02] Palma, N. D., P. Laumay and L. Bellissard, "Ensuring Dynamic Reconfiguration Consistency", SIRAC project, 2002.
- [Rasche05] Rasche, A. and A. Polze, "Dynamic Reconfiguration of Component-based Real-time Software", 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, January 2005.
- [Rumbaugh04] Rumbaugh, J., G. Booch, and I. Jacobson, "The Unified Modeling Language Reference Manual," Second Edition, Addison Wesley, Reading MA, 2004.
- [Shin05] Shin, M. E., "Self-Healing Component in Robust Software Architecture for Concurrent and Distributed Systems," Science of Computer Programming, Vol. 57, No. 1, 2005.
- [Shin06] Shin, M. E. and J. H. An, "Self-Reconfiguration in Self-Healing Systems," 3<sup>rd</sup> IEEE Workshop on Engineering of Autonomic and Autonomous Systems (EASe 2006), Columbia, MD, USA, 25-27 April 2006.



# Task Decomposition for Adaptive Data Staging in Workflows for Distributed Environments

Onyeka Ezenwoye<sup>1</sup>, Balaji Viswanathan<sup>4</sup>, S. Masoud Sadjadi<sup>2</sup>, Liana Fong<sup>3</sup>, Gargi Dasgupta<sup>4</sup>, and Selim Kalayci<sup>2</sup>

<sup>1</sup> South Dakota State University, Brookings, SD, USA, onyeka.ezenwoye@sdstate.edu

<sup>2</sup> Florida International University, Miami, FL, USA, sadjadi,skala001@cs.fiu.edu

<sup>3</sup> IBM Watson Research Center, Hawthorne, NY, USA, llfong@us.ibm.com

<sup>4</sup> IBM India Research Lab, New Delhi, India, bviswana,gdasgupt@in.ibm.com

**Abstract**—Scientific workflows are often composed by scientists that are not particularly familiar with performance and fault-tolerance issues of the underlying layer. The inherent nature of the infrastructure and environment for scientific workflow applications means that the movement of data comes with reliability challenges. Improving the reliability scientific workflows in distributed environments, calls for the decoupling of data staging and computation activities, and each aspect needs to be addressed separately

In this paper, we present an approach to managing scientific workflows that specifically provides constructs for reliable data staging. In our framework, data staging tasks are automatically separated from computation tasks in the definition of the workflow. High-level policies can be provided that allow for dynamic adaptation of the workflow to occur. Our approach permits the separate specification of the functional and non-functional requirements of the application and is dynamic enough to allow for the alteration of the workflow at runtime for optimization.

**Keywords:** Data Staging, Scientific Workflow, Distributed Systems.

## I. INTRODUCTION

In a distributed computing system, components may reside in different physical locations. These components are often encapsulated as self-contained and internet-accessible software components or applications. Distributed applications are exposed as reusable components that can be dynamically discovered and integrated to create new applications. These new applications form aggregate (or composite) services. In this composite model, the composite application is an aggregation of tasks that are performed/executed by the integrated distributed applications, as illustrated in Figure 1.

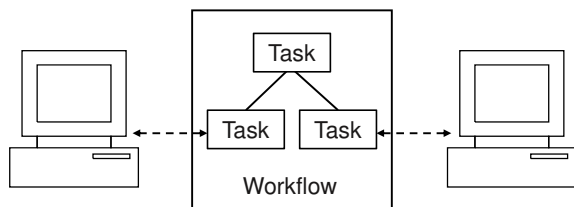


Fig. 1. Workflows aggregate tasks that get executed on the distributed computers

In the research community, these distributed applications and the ability to aggregate them is allowing scientist to create

complex scientific applications. To facilitate this integration are computing infrastructures such as the Grid [6] that allow for the harnessing of resources available on disparate distributed computing environments to create a parallel infrastructure that allows for applications to be processed in a distributed manner. These applications, which are geared towards scientific discovery tend to be compute and data intensive, requiring large amounts of data to be moved around the system. Since moving the application close to the data is not always practical due to insufficient computational resources at the storage site [8], data needs to be moved to the applications that need them and in some cases cleanup operations need to occur after application execution. The inherent nature of the infrastructure and environment for these applications means that the migration of data comes with certain challenges. The successful execution of applications is dependent of the availability of necessary data. For instance, workflow mapping techniques may produce workflows that are unable to execute due to the lack of the disk space necessary for the successful execution [10], requiring that data movement be scheduled and monitored. In fact, the magement of data is essential through the entire lifecycle of the workflow from creation to execution, and result management [4]

Scientific workflows are often composed by scientists and as such are not particularly tuned for performance and fault-tolerance [3] This is because workflow languages permit the abstraction of language semantics at a level that is easy for domain specialist to use, thus focus is often placed on the functional aspects of the workflow. Also, since the eventual execution resources are not known during composition, optimizing the runtime of the overall workflow becomes a big issue [3] Although data is a key component in scientific workflow, a lot of emphasis is not placed on providing fault tolerance for tasks related to their data requirements. Data staging tasks are often embedded in computation-related tasks and reliability efforts are then focused on the computation tasks even though data access presents the main bottleneck for data-intensive applications [8]

Improving the reliability of data placement in distributed environments, calls for the decoupling of data movement and computation activities, each aspect needs to be addressed separately [9] Infrastructure for distributed scientific applications needs to consider data movement as part of the end-to-end

performance of the system and care must be taken to make sure they complete successfully and without any need for human intervention [8]

In this paper, we present an approach to managing scientific workflows that specifically provides constructs for reliable data staging. In our framework, data staging tasks are automatically separated from computational tasks in the definition of the workflow. High-level policies are provided that allow for dynamic adaptation to occur. Recovery actions are applied separately for either data or computation-related tasks, for failures that could arise from software, network or storage system. Our approach permits the separate specification of the functional and non-functional requirements of the application and is dynamic enough to allow for the alteration of the workflow at runtime for optimization.

The rest of this paper is structured as follows. In section II presents the architecture of our adaptive workflow manager that decomposed data staging and computation. Section III contains some related work. Finally, some concluding remarks are provided in Section IV.

## II. WORKFLOWS IN GRID ENVIRONMENTS

In this section we present a brief overview of the overall architecture of our workflow management system for grid environments. As part of the Latin American Grid (LA Grid) [2], we have developed a distributed architecture that is comprised of two main middleware components: the workflow manager and the meta-scheduler. The LA Grid model is an end-to-end, layered architecture that is comprised of five main layers (see Figure 2): the Application Layer, which models the business logic of the a complex application in a workflow; the Workflow Management Layer, which enacts the business logic of the workflow and is responsible for maintaining concurrency and sequencing among tasks (or jobs) in the workflow; the Meta-Scheduling Layer, which is responsible for resource selection and job execution control; the Local Resource Management Layer, which is responsible for scheduling and executing individual jobs on the local resources; and the Resource Layer, which is comprised of the actual computing, storage, and networking resources.

To express the workflows themselves, we chose the Web Services Business Process Execution Language (BPEL) [5], which has emerged as the standard workflow language for orchestrating service-based applications. Several production-level software provide core BPEL engines. These engines are virtual machines that interpret and execute BPEL grammar. The grammar models the business logic of the workflow as a directed-graph, where the nodes represent tasks and the edges represent inter-task dependencies, data flow or flow control.

Currently, the BPEL specification does not contain the necessary semantics or support for defining jobs. Grid jobs require the richness and flexibility for specifying varied resource requirements and system environments. The Open Grid Forum job scheduling working group recommends the use of Job Submission Definition Language (JSDL) [1], for capturing a job's resource and environment requirements as well as

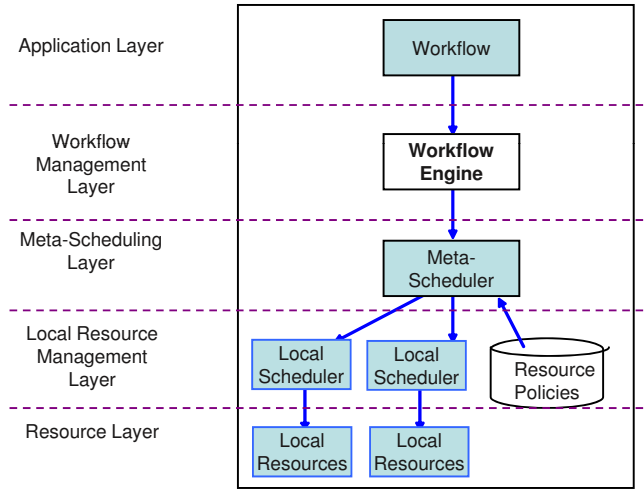


Fig. 2. A layered architecture for workflow execution in grid environments

data dependencies. In absence of unified modeling support, BPEL and JSDL are used to provide the combined modeling semantics for the workflow. This way individual workflow tasks are represented as JSDL jobs, embedded with BPEL constructs. This provides the necessary environment based on standardized technologies.

### A. Data Staging

Many Grid jobs require input data, and in the absence of a shared file system, these datasets need to be staged in at the site of execution. Usually the data stage-in needs to be completed before the job can begin execution. In workflows, the data requirement could be an input to the system or produced by the execution of a preceding job. In the latter case, a data-dependency is created in the flow between the producer and the consumer jobs of the data. Thus a typical data staging pattern in workflows comprises of a data stage-in from either producer jobs or from defined inputs, followed by a job submission pattern. In some cases, a data stage-out is specified to perform data cleanup operations after execution. There may be several such data staging activities, which could occur sequentially or in parallel.

Once the data staging of all dependencies are satisfied, a job can be submitted for execution. Typically, data staging activities are embedded with the specification of the computation task in the JSDL document, as illustrated in Figure 3 This complex JSDL is then submitted as one job submission request by the workflow. Within this framework, it becomes difficult to isolate the source of failures. For instance, faults generated by data stage-in can get propagated to job execution.

### B. Decoupling Data Staging

In this section, we present our approach to providing an adaptive workflow execution. Our approach takes into consideration the need to provide adaptive data staging as part of the end-to-end performance of the workflow, by automatically decoupling the specification of data staging and computation.

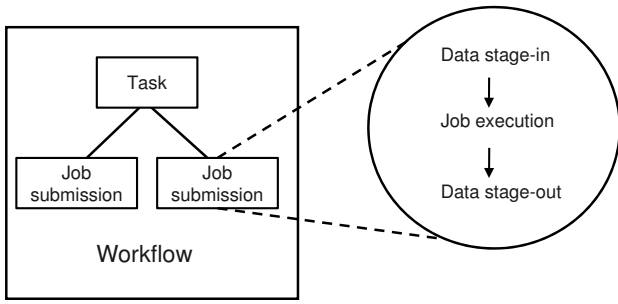


Fig. 3. A job submission task with embedded data staging activities.

Not decoupling data staging and computation affects capability of system to provide fault-tolerance and adapt to environment and user preferences. Data staging jobs and computational jobs need to be differentiated from each other within the system. Figure 4 illustrates the architecture of our adaptive workflow manager.

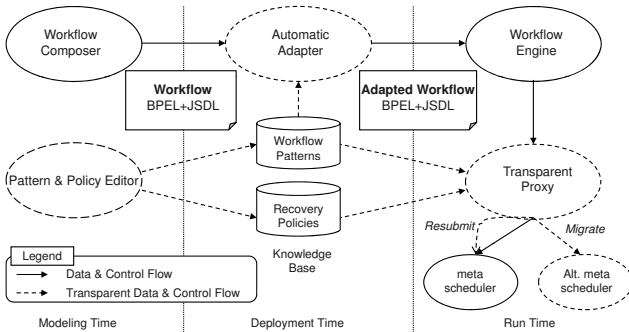


Fig. 4. The architecture of our adaptive workflow manager

In the left side of Figure 4, a domain expert will use the Workflow Composer to specify the business logic of the application using BPEL+JSDL. The domain expert should only be concerned about the business logic of the application and not about handling faults and exceptions. The job descriptions (in JSDL) are treated as XML complex types, which in turn are used as the parameters to some Invoke constructs in BPEL. It is within this JSDL definitions that data staging and computation tasks are encapsulated. During deployment time, the resulting workflow is passed to the Automatic Adapter, which automatically generates a functionally equivalent workflow. It is during this adaptation phase that the complex JSDL definitions that data staging and computation tasks are decomposed in primitive JSDL definition. Separate definitions and invocations are defined in the workflow for data stage-in, computation, and data stage-out. The invocation messages are extended with context information so that correlations can be made between the decomposed tasks. The context information is also needed for the Proxy to monitor the interaction between the workflow manager and the meta-schedulers.

The automatic adapter has an algorithm that identifies the known workflow patterns (e.g. job submission and data staging) within the workflow. The most updated workflow

patterns are stored in the Knowledge Base. New workflow patterns can be added to the knowledge base using the Pattern and Policy Editor. The generated workflow, called adapted workflow, would not include constructs to handle faults at run time. Instead workflow behavior is modified at run time through the Transparent Proxy. At run time, the workflow will be executed by the Workflow Engine. The workflow engine can be any standard BPEL engine, as we did not extend BPEL in our work. During the automatic adaptation of the workflow, all the calls originally targeted for the local Meta-scheduler are redirected to the Transparent Proxy [7]. Therefore, the Transparent Proxy will intercept all the calls to the Meta-scheduler.

The Proxy will appear as a Meta-scheduler to the workflow process, and as a workflow process to the Meta-scheduler; hence, the name transparent. Its main responsibility includes submission of the jobs to the local Meta-scheduler and notifying the workflow process of the job status when it receives job status updates from the Meta-scheduler. In addition, it implements a pattern-matching algorithm that monitors the behavior of the intercepted calls and provides fault-tolerant behavior when faults occur. The algorithm is based on the Recovery Policies, the context information embedded in the adapted workflow, the Workflow Patterns, and their corresponding Fault-Tolerant Patterns. For example, following the recovery policies governing the current faulty situation, the Transparent Proxy may resubmit the job (data staging or computation) to the same Meta-scheduler or migrate it to another Meta-scheduler.

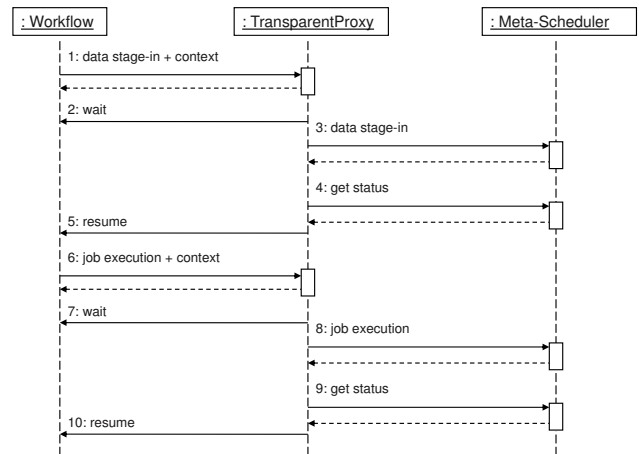


Fig. 5. Sequence diagram showing the interaction between the workflow engine, transparent proxy and meta-scheduler

Figure 5 shows the interaction between the workflow engine, transparent proxy and meta-scheduler. Some messages have been simplified or removed for the purpose of brevity. As depicted in the diagram, data stage-in and job execution submission are decomposed and separated (data stage-out is not shown). Data Stage-in jobs are submitted first to the proxy with some context information that it needs to correlate related job execution and data stage-out submissions. The workflow

is made to wait while the proxy attempts to execute the data stage-in task. The proxy will apply recovery actions (such as retry or migrate) based on specified high-level policies. Upon successful data staging, the workflow is allowed to proceed and further tasks can be submitted.

### III. RELATED WORK

The work by Chervenak [3] is concerned with data placement policies that distribute data in ways that are advantageous for application execution, for instance, by placing data sets so that they may be staged into or out of computations efficiently or by replicating them for improved performance and reliability. Their work centers on prestaging data using the Data Replication Service versus using the native data stage-in mechanisms of the Pegasus workflow management system. A policy-driven data placement service is responsible for replicating and distributing data items in conformance with policies or preferences. This work differs from ours because it applies data management techniques at the local resource management layer (see Figure 4), while our work focuses on the workflow management layer.

Kosar [8] presents a data placement subsystem that allows for data for distributed computing systems to be queued, scheduled, monitored, managed, and checkpointed. Their framework includes a specialized scheduler for data placement, a high level planner aware of data placement jobs, a resource broker/policy enforcer and optimization tools. Data placement jobs are represented in a different way than computational jobs in the job specification language so that the high level planners can differentiate these two classes of jobs. The system can perform reliable data placement, and recover from failures without any human intervention. This work does is not dynamic since it requires that the workflow definition be modified and redeployed in order for any adaptation to occur. In comparison, our approach is automatic and includes context information for better fault tolerance. Also by focusing on the workflow management layer, we assume no control over data and job scheduling.

Ranganathan's [9] framework allows for data movement operations may be tightly bound to job scheduling decisions or, performed by a decoupled, asynchronous process on the basis of observed data access patterns and load. A scheduling framework within which a wide variety of scheduling algorithms can be used. They assume a multi-user and multi-site model. At each site, there are 3 components: an External scheduler; a local scheduler; and a Dataset scheduler. This work differs from ours because it considers data management issues at the local resource management layer, while our work focuses on the workflow management layer.

Singh [10] focuses on optimizing disk usage and scheduling large-scale scientific workflows onto distributed resources. Their approach is minimize the amount of space a workflow requires during execution by removing data files at runtime when they are no longer needed. To achieve this, workflows are restructured to reduce the overall data footprint of the workflow. Their algorithms adds a cleanup job (data stage-out)

for a data file when that file is no longer required by other tasks in the workflow. Similar to our approach, their workflow adaptation algorithm is applied after the executable workflow has been created but before the workflow is executed. However, the issue of fault-tolerance is not addressed and no data cleanup if a compute task fails.

### IV. CONCLUSION

In this paper, we present an approach to managing scientific workflows that specifically provides constructs for reliable data staging. In our framework, data staging tasks are automatically separated from computational tasks in the definition of the workflow. High-level policies can be provided that allow for dynamic adaptation of the workflow to occur. Recovery actions are applied separately for either data or computation-related tasks, for failures that could arise from software, network or storage system. Our approach permits the separate specification of the functional and non-functional requirements of the application and is dynamic enough to allow for the alteration of the workflow at runtime for optimization.

**Acknowledgement:** This work was supported in part by IBM, the National Science Foundation (grants OISE-0730065, OCI-0636031, HRD-0833093, and IIP-0829576). Opinions expressed in this material are those of the author(s) and not the NSF and IBM.

### REFERENCES

- [1] A. Anjomshoaa, A. Anjomshoaa, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. *Job Submission Description Language(JSDL) Version 1.0*, November 2005.
- [2] R. Badia, G. Dasgupta, O. Ezenwoye, L. Fong, H. Ho, Y. Liu, S. Luis, A. Praino, J.-P. Prost, A. Radwan, S. M. Sadjadi, S. Shivaji, B. Viswanathan, P. Welsh, and A. Younis. Innovative grid technologies applied to bioinformatics and hurricane mitigation. In *High Performance Computing and Grids in Action*. IOC Press, 2007.
- [3] A. Chervenak, E. Deelman, M. Livny, M.-H. Su, R. Schuler, S. Bharathi, G. Mehta, and K. Vahi. Data placement for scientific applications in distributed environments. *IEEE/ACM International Workshop on Grid Computing*, 2007.
- [4] E. Deelman and A. Chervenak. Data management challenges of data-intensive scientific workflows. In *Proceedings of the Eighth IEEE International Symposium on Cluster Computing and the Grid*, Washington, DC, USA, 2008. IEEE Computer Society.
- [5] O. Ezenwoye and S. M. Sadjadi. Composing aggregate web services in BPDL. In *Proceedings of The 44th ACM Southeast Conference*, Melbourne, Florida, March 2006.
- [6] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2001.
- [7] S. Kalayci, O. Ezenwoye, B. Viswanathan, G. Dasgupta, S. M. Sadjadi, and L. Fong. Design and implementation of a fault tolerant job flow manager using job flow patterns and recovery policies. In *Proceedings of the 6th International Conference on Service Oriented Computing*, Sydney, Australia, December 2008. Springer Berlin.
- [8] T. Kosar and M. Livny. A framework for reliable and efficient data placement in distributed computing systems. *Journal of Parallel and Distributed Computing*, 2005.
- [9] K. Ranganathan and I. Foster. Decoupling computation and data scheduling in distributed data-intensive applications. In *International Symposium on High-Performance Distributed Computing*, Los Alamitos, CA, USA, 2002. IEEE Computer Society.
- [10] G. Singh, K. Vahi, A. Ramakrishnan, G. Mehta, E. Deelman, H. Zhao, R. Sakellariou, K. Blackburn, D. Brown, S. Fairhurst, D. Meyers, B. Berriman, J. Good, and D. Katz. Optimizing workflow data footprint. *Scientific Programming*, 15(4):249–268, 2007.

# Constructing FODA Feature Diagrams with a GUI-based Tool

Shin NAKAJIMA  
National Institute of Informatics  
Tokyo, Japan  
nkjm@nii.ac.jp

## Abstract

*Although Feature Diagram proposed by K. Kang et al is intuitive and easy to understand, two issues manifest themselves when drawing diagrams of non-trivial size; supporting construction, and checking consistency. FD-Checker provides a GUI solution to the two issues. The tool is implemented by customizing iDot. The checking method follows the idea of representing a feature diagram in terms of propositional logic formula. The formal analysis is a problem of satisfiability checking, which can be performed automatically with Alloy.*

## 1. Introduction

Feature-Oriented Domain Analysis (FODA), proposed by K. Kang et al in 1990 [9], is a method used in Software Product Line Engineering (SPLE) [4][15]. It focuses on identifying and classifying *features* of product family at the early stage of system development.

FODA employs *Feature Diagram* as its concrete modeling notation [9]. It is basically an acyclic AND-OR graph; some of the nodes are common features and the others are variabilities. The common features are included in all the software products considered, while the variabilities are implemented only in some of the products. Namely, a product consists of all the common features and may have some variabilities. Since SPLE concerns about a product family as a whole, but not about an individual product, identifying variabilities is the most important activity in FODA.

Since it is diagram-based, feature diagram is intuitive and easy to understand. We, however, are faced with two issues when drawing feature diagrams of non-trivial size.

- (1) Supporting construction,
- (2) Checking consistency.

In order to address the first issue, a GUI-based editor would be needed, which helps users draw large feature diagrams in

a simple way. As for the second, certain formal semantics are to define for the feature diagram, which becomes a basis for checking consistency of drawn diagrams.

This paper presents *FD-Checker*, which is a GUI-based editor and is, at the same time, a GUI front-end for an automated checker. *FD-Checker* is implemented by customizing *iDot* [1] in order to readily include fancy displaying functions, such as interactive resizing or fish-eye view. The formal analysis method follows the idea of representing a feature diagram in terms of propositional logic formula, and then the analysis is a problem of satisfiability checking. Our method employs Alloy [7] as the engine for the automatic analysis. *FD-Checker* provides a GUI solution to the above two issues.

The paper is structured as follows. Section 2 introduces FODA feature diagram. Section 3 presents *FD-Checker*, its tool architecture and discusses how *FD-Checker* supports users to construct feature diagrams. Section 4 describes our approach to formalizing the feature diagram and using Alloy tool for the analysis. Section 5 compares related work and Section 6 concludes the paper.

## 2. FODA Feature Diagrams

We first introduce graphic representation of feature diagrams and their informal meanings. Although a lot of variations have been proposed, the diagrams here are adapted from the one in [5], which is also used in other literatures [6][16] and thus can be considered as a minimum standard. The diagram components are grouped into six primitive types as in Figure 1, which provides means to represent the logical relationships among features essentially to form a tree. Two composition rules in Figure 2 can add further logical constraints between features already existing.

A simple example [6] is shown in Figure 3. The root *Mobile Phone* feature has three direct sub-features. *Earphone* is optional, while either *MP3* or *Camera* or both are mandatory. A side composition rule, depicted as a dotted line, illustrates that *Earphone* and *MP3* are mutually dependent, which means that both are selected or neither. The

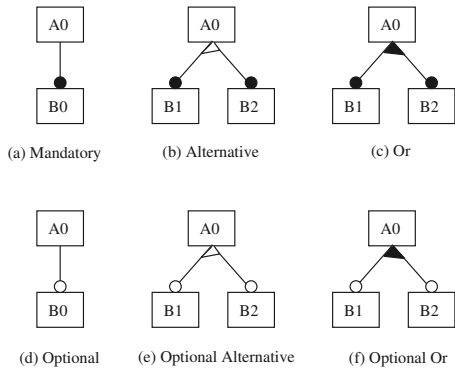


Figure 1. Six Primitives

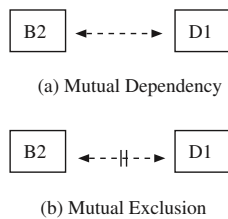


Figure 2. Two Composition Rules

feature diagram shows that MobilePhone feature is common to all the system in the product line of interest, and that the other three are considered as the variabilities. A particular product may have Camera as well as MobilePhone feature.

### 3. FD-Checker

FD-Checker is a GUI-based editor and also provides GUI front-end for the back-end automated analysis tool. We have decided to customize *iDot* tool [1] for the implementation. The *dot* language of *Graphviz*, which *iDot* accepts, is designed to represent many types of directed graphs. By introducing some of new attributes, we can represent the feature diagram in the dot format. Then, we readily make use of fancy display methods such as fish-eye view, interactive resizing, and folding/unfolding of sub-graphs that *iDot* provides. These methods are generally usable for displaying large graph structures. For example, the fish-eye view shows the point of interest in detail and the overview of the graph in the same window by distorting the graphical image.

Figure 4 shows the tool architecture of FD-Checker. In addition to the basic framework provided by *iDot*, it adds three new sub-components, GUI Editor, PO Generator and Extractor. Dot File contains logical rela-

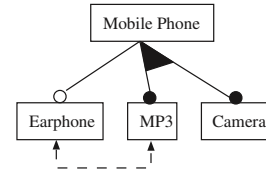


Figure 3. A Simple Example taken from [6]

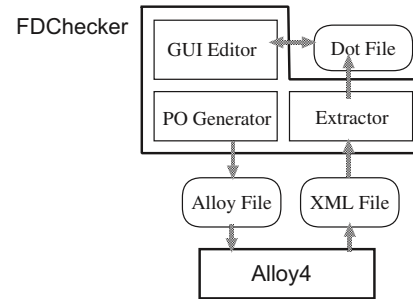


Figure 4. Tool Architecture

tionships for a feature diagram. For example, the feature diagram in Figure 3 is described as below.

```

digraph {
  ...
  33 [label="Mobile Phone",orEdgeGroup="e37,e38",
    must];
  34 [label="Camera"];
  35 [label="MP3"];
  36 [label="Earphone"];
  33 -> 34 [name="e37",optionType="mandatory"];
  33 -> 35 [name="e38",optionType="mandatory"];
  33 -> 36 [name="e39",optionType="optional"];
  35 -> 36 [name="e40",mutualType="dependency",
    startArrow,endArrow];
}

```

GUI Editor provides some standard look-and-feel style editing commands. Figure 5 is a screen snapshot taken when two features, MP3 and Camera, are linked to Mobile Phone as Mandatory Or sub-features.

FD-Checker encourages a *bottom-up* style construction



Figure 5. Editing

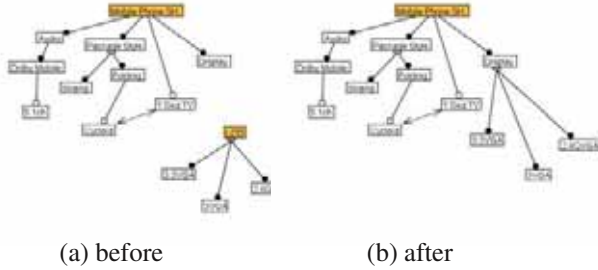


Figure 6. Introducing New Sub-tree

of feature diagrams with the *Fusion* command. One may first find a certain feature and dwell on whether it is elaborated into variabilities; the feature is appropriate to separate variabilities from common features. The resultant features may form a small sub-tree isolated from the main diagram. Since the sub-tree is its part, the root of the sub-tree is *fused* into a place holder in the main.

The example in Figure 6 refers to a family of mobile phones, and the product series may vary in its display size depending on what other features are included. The small sub-tree in the left figure shows that *Display* feature is a variability expanded into three variabilities. *LCD* has, as its sub-features, *3.3VGA*, *3VGA*, and *2.8QVGA*, to be Mandatory-Alternative. The main feature diagram with the root *MobilePhoneSH* refers to a place holder *Display* Mandatory feature. *LCD* is inserted at the location of *Display* by using *Fusion* command.

Since we construct feature diagrams incrementally and interactively, it is not clear during the construction process whether the diagram as a whole is consistent or not. An automatic consistency checking is inevitable. The details will be discussed in Section 4.

*PO Generator*, standing for Proof Obligation Generator, and *Extractor* together provide the interface to the back-end analysis engine *Alloy*. *PO Generator* translates the dot format into *Alloy* source fragments (see Section 4.2). *Extractor* is responsible for extracting the configuration information from *Alloy* analysis result in an XML file and generates a new temporal dot format data for graphical displaying. An example is shown in Figure 7, which is a part of the original diagram in Figure 3; *Camera* is not included here.

#### 4. Automated Analysis

This section describes our approach to formalizing the feature diagram and using *Alloy* [7] for the automatic analysis. The discussion is based on our previous work [14].

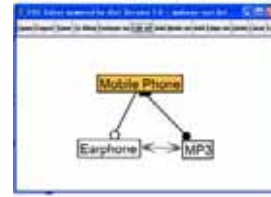


Figure 7. Resultant Configuration

(a) Mandatory	$A0 \Leftrightarrow B0$
(b) Alternative	$(A0 \Leftrightarrow B1 \oplus B2) \wedge \neg(B1 \wedge B2)$
(c) Or	$A0 \Leftrightarrow B1 \vee B2$
(d) Optional	$A0 \Leftarrow B0$
(e) Optional Alternative	$(A0 \Leftarrow B1 \oplus B2) \wedge \neg(B1 \wedge B2)$
(f) Optional Or	$A0 \Leftarrow B1 \vee B2$

Table 1. Encoding in Propositional Logic (1)

#### 4.1. Encodings in Propositional Logic

Firstly, we present formal definitions for the feature diagram. In particular, the method here follows the idea of representing a feature diagram in terms of propositional logic formula. Then the formal analysis is a problem of satisfiability checking.

The encoding method is based on the following observation. Each primitive type (Figure 1) specifies how the sub-feature(s) are included when a super feature  $A0$  is selected. A propositional variable such as  $A0$  is introduced for each feature and understood so that its *true* value means that the feature  $A0$  is selected<sup>1</sup>. The logical constraint relationship posed with each primitive type is represented by a propositional logic formula. Composition rule can add further constraint conditions on features. It provides means to put logical conditions on selected features other than the primitive type rules.

Table 1 and Table 2 summarize their propositional logic interpretation. Each corresponds to the diagrammatic notation in Figure 1 or Figure 2. A complete feature diagram is just a potentially large propositional formula conjoining all the formulas, each corresponding to a primitive type or a composition rule.

Among the formula presented, we here explain the case for Table 1 (a) Mandatory.

(a) Mutual Dependency	$B2 \Leftrightarrow D1$
(b) Mutual Exclusion	$\neg(B2 \wedge D1)$

Table 2. Encoding in Propositional Logic (2)

<sup>1</sup>Its false value stands for discarding the corresponding feature.

The feature B0 is always included when the feature A0 is selected. At the same time, when the feature B0 is known to be included, its super feature A0 is also chosen.

The notion of super/sub feature is not explicitly encoded since we focus on the relationship concerning to whether one feature is to be selected or not when another is selected. The graph structure view is handled with the GUI editor parts of FD-Checker (see Section 3).

Further, we define a well-formed feature diagram to have the following properties.

- (W1) Every feature diagram has only one root feature.
- (W2) No feature becomes one of its own super-features.
- (W3) No feature is an island.

A feature diagram can potentially represent many sets of consistent features. Each consistent set of features, denoting a particular system requirement, is called *Configuration* [10]. A configuration can be defined in a rigorous and compact manner thanks to the propositional encodings.

Given a set of propositional formula  $\Gamma$ , and let each  $\gamma \in \Gamma$  represent a fragment of a feature diagram in propositional logic, a model  $m$  is a truth value assignment for the set of propositional variables to make  $\Gamma$  valid.  $\Gamma$ , here, is understood as a shorthand notation for a formula obtained by conjoining all  $\gamma$ 's. Further,  $\Gamma$  should be augmented with  $\rho$  which stands for the fact that the top root feature propositional variable is always *true* (see (W1)).

$$m \models \Gamma \wedge \rho$$

As the features constituting a configuration have their corresponding propositional variables to be *true*, a configuration  $c$  can be obtained from the model  $m$  by selecting *true* propositions only. Configuration is defined to consist of all the features, that correspond to propositions appearing positively in the model  $m$  of a propositional formula  $\Gamma \wedge \rho$ .

Since a feature diagram as a whole is basically a conjunction of propositional logic formulas, its configuration may be null when the conjunction is unsatisfiable. Such feature diagrams are non-sense in that they do not represent the requirements of any meaningful systems. It is necessary to check whether a given feature diagram is consistent.

Consistency checking is a problem of ensuring that at least one model exists satisfying the relationships. It can be done by finding a model of  $\Gamma \wedge \rho$ .

In addition to the consistency checking, it is sometimes necessary to ensure that there exist configurations to include a particular set of specified features. In other words, the specified features are those that are required a priori. Validation checking is to ensure that such combinations are possible as configurations, which can be done in the same manner as the consistency checking. The logical formula is augmented to include a formula representing that some of the

features are specified a priori. Namely, the model relation turns out to be

$$m \models \Gamma \wedge \rho \wedge \Psi$$

where  $\Psi$  denotes a set of propositions, each of which stands for the fact that a given feature propositional variable is always *true*. Further  $\Psi$  may have negative literals that means to take *false* values, in which case the features corresponding to the specified *false* propositional variables are not included.

## 4.2. Analysis with Alloy

The basic method of checking feature diagrams is a problem of checking satisfiability or finding models of a propositional logic formula. The method can readily be realized with an existing tool Alloy [7].

Alloy is an automatic analysis tool for first-order relational logic formula, developed by D. Jackson at MIT. Alloy compiles a given source description into a propositional logic formula, and uses an external SAT solver to check satisfiability by searching for its model. Alloy uses a bounded search method to achieve decidability for analyzing formula in first-order relational logic.

The bottom-line of the method is just to translate the logical constraints shown in Table 1 and Table 2 into the equivalent Alloy source fragments. Naive encodings, however, sometimes result in under-constraint specifications, for which Alloy returns the analysis results that are not intuitive. The encoding method here adds stronger constraint conditions than the naive approach, in that selected features are to be linked to their super feature. In other words, the super-sub feature relationships are explicitly encoded. Although some of the informations are redundant, the description is neither under-constraint nor over-constraint in view of the analysis with Alloy.

The details of the encoding are explained in order. Firstly, a feature is defined as an abstract signature *Feature*. It has two fields *up* and *select*; *up* refers to its super feature being of its type *Feature*, and *select* is a marker to show whether the feature is selected.

```
abstract sig Feature
    { up : Feature, select : Selected }
fact Suppress { all f : Feature |
    (f.select = No) => no(f.up) }
fact NoCycle { no f : Feature | f in f.^up }
```

In Alloy, a signature represents a set of atoms. An abstract sig, however, is not considered to have atoms. It represents a classification of elements being further refined by concrete signatures. Each concrete signature refers to a feature appearing in the feature diagram.



Two global constraints are imposed on `Feature`. `Suppress` specifies that a feature not selected in the configurations has no super feature, which is effective to avoid under-constraint situations. `NoCycle` corresponds to the rule (W2).

The above `Feature` and the two global constraints constitute the basic description framework. Each concrete feature is defined as a singleton (`one sig`) signature extending `Feature`. A newly defined one becomes a subset of its parent abstract `Feature`. For example, below shows a fragment; `MobilePhone` is the root and `Camera` is another feature. A root feature is special in that it has no super (`no up`).

```
one sig MobilePhone extends Feature {}{ no up }
one sig Camera extends Feature {}
```

Secondly, each primitive type or composition rule is translated into a global constraint `fact`. During the analysis, the specified conditions are always respected and the model finding is performed while satisfying all of them. Below shows `fact` descriptions for a template of Mandatory (Table 1 (a)).

```
fact {
  (A0.select = Yes)
  => (B0.select = Yes) and (B0.up = A0)
  else (B0.select = No)
}
```

A Mandatory type specifies that both `A0` and `B0` are selected or neither. Further, `A0` becomes the super feature of `B0` when they are selected. This informal statement is encoded into the above `fact` formula, using an Alloy syntax of `_=>_else_` for `if_then_else_`, and `and` for a logical and  $\wedge$ .

Since the above formula looks awkward, we may use syntactic macros<sup>2</sup> to improve the readability.

```
fact { Require(A0) => Choose(B0, A0)
      else Ignore(B0) }
```

The composition rules (Table 2) are similarly translated into `fact` formulas. Since the rules are meant to put further constraints on features, they do not access the `up` field and thus `Choose` is never used. For example, a mutually-dependent rule in Table 2(b) is represented as below.

```
fact { Require(B2) <=> Require(D1) }
```

All the instantiated `fact` and the user-defined signatures together with the basic framework declarations constitute an Alloy description of a given feature diagram.

Thirdly, consistency checking is conducted by using Alloy `run` command where the condition is so specified that the root feature is always selected (see (W1)). Note that

<sup>2</sup>A good old cpp macro processor is used.

`run` command here is not accompanied with search scope parameters. In the proposed encoding, each feature is declared as a singleton (`one sig`) and the number of atoms in a set to search for the model is known.

```
run { Require(MobilePhone) }
```

Validation checking can be done similarly, but `run` command may have further elements. For example, the command below is used for a check whether a configuration, containing both `MobilePhone` and `MP3` features, exists. In the command, `Require(MobilePhone)` and `Require(MP3)` correspond to  $\rho$  and  $\Psi$  respectively.

```
run { Require(MobilePhone) and Require(MP3) }
```

The `run` command may have `Ignore(MP3)` instead of `Require(MP3)` when the user's intention is to *deselect* `MP3` from the resultant configuration.

Last, as shown in Figure 4, `FD-Checker` and `Alloy` exchange information via files. `PO Generator` generates an Alloy source file of the feature diagram in the editor buffer. Alloy initiates its analysis by reading the file and produces its output in an XML file. `Extractor` then reads the XML file and displays the obtained configuration. A solution instance in Figure 7 shows that three features `MobilePhone`, `Earphone`, and `MP3` are selected and form a small tree. It also shows that `Camera` feature is not selected in this particular configuration.

## 5. Related Work

Since FODA is getting widely used, formal definitions of feature diagrams have been studied. A graphical presentation of the feature diagram has its origin in K. Kang et al [9]. While some variations have been proposed, the notation in [5] is often used in the literature [6][16]. This paper also adapts the same notation.

The idea of connecting propositional logic formulas to feature diagrams is due to M. Mannion [13]. J. Sun et al [16] have taken a similar approach to defining formal semantics of the feature diagram, and shown how to use `Z/EVES` and Alloy for reasoning about the properties. D. Batory [2] has followed the idea of M. Mannion to have a logic-based representation of the feature diagram. He employed `LTMS` (Logic-Truth Maintenance Systems) as a back-end engine to search for valid configurations. The search is basically the same as model-finding. D. Benavides et al [3] transform an invalid feature diagram into a CSP (Constraint Satisfaction Problem) in linear integer arithmetics, and solve the problem with `CLP` (Constraint Logic Programming) approach. Their approach can deal with diagrams more expressive than what is discussed in this paper. R. Gheyi et al [6] have employed Alloy to define formal semantics of

the feature diagram. M. Janota and J. Kiniry [8] adapt a higher-order logic prover PVS for the interactive analysis. W. Zhang et al [17] uses a model-checker SMV for the automated analysis of feature diagrams.

The motivation of the formalization, either using an automated analysis tool or not, is to reason about the properties of feature diagram. Especially, the work [6][16] have focused on discussing *re-factorings* of feature diagrams. And thus the encoding is more like a *deep embedding*, in which the semantics of primitive types and composition rules are explicitly formalized in the host specification language or logic.

The encoding in this paper is, instead, a *shallow embedding*, in which a component appearing in a given feature diagram is translated directly into a language element of Alloy. Because of the shallow embedding, our method cannot reason about the properties of diagrams in general. It, however, is suitable for the efficient analysis of given feature diagrams to check both consistency and validation. The diagram can be represented by the formula shorter than the case of deep encoding methods. Further, in our method, run command is not accompanied with search scope parameters, which simplifies the situation. Alloy usually requires the parameters to achieve the bounded search, and the choice is a kind of an *art* since it affects the analysis results.

The efficiency of the analysis is satisfactory. A medium size feature diagram consisting of 33 features can be analyzed in 4 seconds by Alloy4 on WindowsXP. It reduces to 0.18 seconds if we omit `NoCycle` check. The property (W2) may be ensured at the time of constructing feature diagrams, and thus can sometimes be omitted.

As mentioned above, some work presents methods of using automated analysis with Alloy [6] [16] or SMV [17]. However, they have not develop further tools like FD-Checker to address the issues on supporting the construction of feature diagram. ASADAL [11] is an integrated environment for SPLE-based software development. It provides supports for the FODA-style domain analysis, but does not put emphasis on the formal checking of feature diagrams.

## 6. Conclusion

We have implemented a concept demo tool *FD-Checker*, which is a GUI-based tool for constructing FODA feature diagrams. The tool is implemented by customizing *iDot* [1] so that we can readily use fancy displaying functions, such as interactive resizing or fish-eye view. The formal analysis method in the paper follows the idea of representing a feature diagram in propositional logic formula, and the analysis is just a problem of satisfiability checking. The method employs *Alloy* [7] as the engine for the automatic analysis.

Our hypothesis is that a bottom-up approach for identifying variabilities is better than a top-down approach with the decomposition of the root feature [14]. The disadvantage of the top-down approach has also been recognized in [12]. However, it is only through experience to reach a definitive answer. We will plan to use *FD-Checker* for accumulating further experience on how engineers in industry conduct their work of constructing *good* feature diagrams.

## References

- [1] iDOT/prefuse Web Site. <http://prefuse.org>.
- [2] D. Batory. Feature Models, Grammars, and Propositional Formulas. In *Proc. SPLC 2005*, pages 7–20, 2005.
- [3] D. Benavides, P. Trinidad, and A. Ruiz-Cortes. Automated Reasoning on Feature Models. In *Proc. CAiSE'05*, pages 491–503, 2005.
- [4] P. Clements and L. Northrop. *Software Product Lines*. Addison-Wesley 2002.
- [5] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison Wesley 2000.
- [6] R. Gheyi, T. Massoni, and P. Borba. A Theory for Feature Models in Alloy. In *Proc. First Alloy Workshop at FSE*, 2006.
- [7] D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. MIT Press 2006.
- [8] M. Janota and J. Kiniry. Reasoning about Feature Models in Higher-Order Logic. In *Proc. 11th SPLC*, pages 13–22, 2007.
- [9] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-Oriented Domain Analysis Feasibility Study. CMU/SEI-90-TR-21, 1990.
- [10] K. Kang, J. Lee, and P. Donohoe. Feature-Oriented Product Line Engineering. *IEEE Software*, vol.9, no.4, pages 58–65, 2002.
- [11] K. Kim, H. Kim, M. Ahn, M. Seo, Y. Chang, and K.C. Kang. ASADAL: A Tool System for Co-Development of Software and Test Environment based on Product Line Engineering. In *Proc. ICSE'06*, pages 783–786, 2006.
- [12] K. Lee, K. Kang, and J. Lee. Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. In *Proc. ICSR-7*, pages 62–77, 2002.
- [13] M. Mannion. Using First-Order Logic for Product Line Model Validation. In *Proc. SPLC2*, pages 176–187, 2002.
- [14] S. Nakajima and N. Ubayashi. Lightweight Formal Analysis of FODA Feature Diagrams. In *Proc. RISE 2007*, pages 3–18, 2007.
- [15] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering*. Springer 2005.
- [16] J. Sun, H. Zhang, Y. Li, and H. Wang. Formal Semantics and Verification for Feature Modeling. In *Proc. ICECCS 2005*, pages 303–312, 2005.
- [17] W. Zhang, H. Zhao, and H. Mei. A Propositional Logic-Based Method for Verification of Feature Models. In *Proc. ICFEM 2004*, pages 115–130, 2004.

# Towards a Classification of Requirements Relationships

Ruhaya Ab Aziz, Didar Zowghi, Tom McBride

*Faculty of Engineering and Information Technology, University of Technology, Sydney,  
Australia*

*{raaziz, didar, mcbride}@it.uts.edu.au*

## Abstract

*Requirements are related to and affect each other in many different ways. Developing a comprehensive knowledge of these relationships is an important part of understanding requirements. This paper proposes a classification of requirements relationships from several perspectives such as Feature Oriented, Aspect Oriented and Goal Oriented Approaches. We compare and contrast these relationship classifications and provide examples of each to increase our understanding of this complex phenomenon. This paper aims at integrating requirements relationship classifications from major bodies of work in requirements engineering and to improve awareness on the role they play in software testing practices.*

## 1. Introduction

Requirements may be categorized in many different ways, for example as functional requirements, non-functional requirements, business requirements or user requirements. During software development, these types of requirements are related to one another in several ways. For instance, there may be complex functional relationships between requirements.

A software system may evolve when the environment or stakeholders' requirements change, or for a number of other reasons. Given the potentially complex relationships among requirements, these changes may cause challenging problems in change management for software developers and stakeholders alike. Requirements engineers have to choose the right requirements management techniques very carefully in order to address these challenges. The relationships between requirements should be studied thoroughly when change impacts are being analyzed and before any changes are implemented.

The study of requirements relationships is not new, but there has been little detailed research into the nature of requirements relationships [1]. Carlshamre et

al conducted a survey on requirements relationships in software product release planning in five different companies [2]. The study reports that only 20% of the requirements are singular, which mean most of the requirements, are related to other requirements. Several researchers consider hierarchical relationships between requirements [3-5]. For example, Robertson and Robertson [5] propose non-subjective hierarchy in grouping requirements. Using this approach, requirements specification is organized in a hierarchy where the work context is at the highest level while atomic requirement is at the lowest. It seems that hierarchical relationship is perhaps one of the most common ways to classify requirements relationships.

Research on requirements relationships has extended beyond hierarchical relationships in recent years. Feature Oriented approaches [6-9], Goal Oriented approaches [10-12] and Aspect Oriented approaches [13, 14] have also been investigated. In addition, Robinson et al introduced Requirement Interaction Management (RIM) to deal with conflicts in requirements relationships [15]. However, identifying and managing requirements relationships are still being reported as a problem [16] while studies that explore common characteristics in requirements relationships such as those covered in [1] are still limited.

In this paper we will develop a classification of requirements relationships. We also give examples of each relationship type to help improve our understanding of the nature of these complex relationships. Understanding and documenting these complex relationships are important as they lead to a more effective management of requirements during software development. Thus, this classification aims at summarizing the current literature on classification of requirements relationships by proposing answers to the following questions:

1. What types of requirements relationships are proposed in the literature?
2. What is the nature of these relationships between requirements?

3. How can we utilize the knowledge of these relationships in software development especially for software testing practices?

This paper is organized as following: Section 2 provides an overview of the related research in requirements relationships. Section 3 discusses classification of Requirements relationships. Section 4 provides some discussions on the use of requirements relationships classification in practice. Future research is then discussed in the conclusion section.

## 2. Related Research

Dahlstedt and Persson [1] provide an excellent introduction on requirements interdependencies and construct a fundamental model of interdependency classifications from their literature survey and several interviews. In contrast, Carlshamre et al [2] create the interdependency classifications as a part of an in-depth study on requirement interdependency characteristics and applications in software product release planning. Davis [17] discusses the relationships classification and their application in requirements triage from his years of experience conducting research into requirements engineering in industry and academia. Studies in the Feature Oriented approach discuss the classification of requirement dependency using a feature as a set of tightly related requirements [6-9]. Also, the Goal Oriented approach uses the concept of requirement relationships to represent the relationship between Goals and Sub-goals; Goals and Agents; and several different Goals [10-12]. Finally, the relationships between requirements are represented in Crosscutting Concerns in Aspect Oriented approach [13, 14].

## 3. Classification of Requirements Relationships

Classification is an effective technique for improving our understanding about phenomena of interest. The fact that most requirements are not independent but related to each other is well known and has been illustrated by previous studies (e.g. [1, 2, 17]). The classifications developed in these studies vary and sometimes overlap which may be because the classifications come from different perspectives. We compare and contrast these perspectives and also acknowledge the existing relationships discussed in object oriented method and relational database. In object oriented method, objects and classes and their relationships represent a static view of a system [18]. This means that relationships between objects and classes are static and can directly be implemented in

programming (i.e. object oriented programming). In contrast, requirements need to be modeled with a focus on what the users need to do with the system and the functionality it must contain, and not concerned with how it will be constructed.

Our preliminary classification of requirements relationships aims to integrate and combine previously developed classifications from diverse perspectives. Our proposal of five requirements relationship types is developed by conducting detailed thematic analysis over several related studies from various perspectives. Using thematic coding, requirements relationships can thus be classified into a number of broad categories: *Structural*, *Implementation*, *Temporal*, *Causality*, and *Necessity*. These categories are described in the following sections.

### 3.1. Structural

Structural relationships are relevant where the relationships can be organized and characterized by their structure. Some studies address this relationship as Hierarchy [4] and others as Static [7]. There are three classes of Structural relationships: *refinement*, *is-a*, and *aggregation*:

- **Refinement** - In this relationship, a higher level requirement is refined or elaborated by a number of detailed requirements. For instance, as illustrated in Figure 1 for a course registration system, requirement 1 is refined by requirements 2 and 3. In Feature Oriented, there is one relationship which fall into this category: *Characterization* [8] which is similar to *Refined to* introduced by Dahlstedt and Persson [1] and *Cover*[17]

Requirement 1: The system shall enable students to register for courses in two conditions
Requirement 2: The system shall enable the registration for current semester
Requirement 3: The system shall enable the registration for registered student only

**Figure 1: Example of Structure (refinement) relationship type**

- **'Is a'** - Is-a (inheritance) hierarchy is used to define how a parent requirement has various links to a number of child requirements and vice versa.. This is similar to subtype hierarchy introduce in relational database [19]. Two relationships can be classified under this classification: *Specialization* [7, 8] and *Generalization* [7, 9]. For example, as illustrated in Figure 2 requirement 1 is satisfied when requirement 2 or 3 are satisfied and vice versa.

Requirement 1: The system shall enable student to pay registration fee
--

Requirement 2: The system shall enable student to pay registration fee using cash card  
 Requirement 3: The system shall enable student to pay registration fee using credit card

**Figure 2: Example of Structure (is-a hierarchy) Relationship Type**

- **Aggregation** – In this category, complex requirements are broken down into their components, identifying simpler requirements which describe whole-part hierarchy. There are a few relationship types in the literature that can be classified in this category. In Feature Oriented, there are two dependency relationship which fall into this category: *Composition* [9] and *Decomposition* [7, 8] which are similar to *And refinement* in Goal Oriented [11] and *Subset* introduced in [17]. According to Figure 3, the functions of requirements 1.1 and 1.2 can be satisfied by requirement 1 but not vice versa.

Requirement 1: The system shall enable student register for courses by providing several services  
 Requirement 1.1: The system shall provides a list of all courses offering  
 Requirement 1.2: The system shall provides a module to enable student to modify or delete course selection

**Figure 3: Example of Structure (Aggregation) Relationship Type**

### 3.2. Implementation

Implementation relationships are concerned with the implementation of an application / system / software project. A decision to implement a set of requirements may affect the implementation of another set of requirements in various ways:

- **Value related** - A decision to choose a requirement for implementation may affect the value to the customer of implementing another requirement positively or negatively. For instance, in Figure 4 a decision of implementing requirement 1 may typically decrease the value of a reference book that a student has to buy from a book shop. This kind of relationship has been introduced by many researchers (e.g. [1, 2, 6, 17]). Buhne et al [6] classify the relationship as *Hints* and *Hinders* to show how the implementation of one requirement influences the value of another requirement positively and negatively. Similarly, *Increased /Decreased value* dependency is proposed in [1], *Cvalue* in [2] and *Value* dependency in [17].
- **Cost related** - A decision to choose a requirement for implementation may affect the cost and effort of implementing another requirement. For example, requirement 2 in Figure 4 will be satisfied without using much effort and cost if we have met requirement 3. This kind of relationship

has been addressed by several researchers such as *Icost* in [2], *Increased/ Decreased Cost* in [1] and *Effort* in [17].

- **Conflict** - A decision to choose a requirement for implementation gives a negative influence to another requirement where both requirements cannot concurrently exist for implementation. It also means that the increasing satisfaction for one requirement can decrease the satisfaction of another. For example, requirement 4 in Figure 4 is in conflict with requirement 5 and both of them cannot be implemented at the same time. It is interesting to see that various perspectives have addressed this kind of relationship but by using another name. In Features Oriented, *Excluded* dependency [9] and *Exclusive* dependency [6] are relatively similar to *Disabling* [14] which was introduced in Aspect Oriented and that of *Conflict With* relationship in [1]
- **Similar** - A requirement is similar to or overlaps with one or more requirements in terms of the expression used and the idea of how the requirements will be implemented. This relationship also means that only one requirement needs to be implemented at the same time. This relationship is addressed as *Similar\_to* [1], *Or* [2] and also *Or\_refinement* in Goal Oriented approach [11]

Requirement 1: All reference books shall be provided online.  
 Requirement 2: The system shall provide list of students who have registered for a course  
 Requirement 3: The system shall provide list of students registered for all courses  
 Requirement 4: The system shall enable professor to select one or more courses offering to teach  
 Requirement 5: Only dean has the authorization to select professor to teach the courses offering.

**Figure 4: Example of Implementation Relationship Type**

### 3.3. Temporal

Temporal relationship is concerned with actions in a specified temporal order. This kind of relationship will also capture the various ways that requirements are related in real time applications. There are a number of situations that can describe this category:

- A requirement should be implemented immediately before or after another. This relationship typically represents a pre condition and/or post condition. In Figure 5, Requirement 2 has to be implemented before Requirement 3. This kind of relationship is introduced as *serial dependency* [7] in Feature Oriented, *Enabling* [14] in Aspect oriented and *Temporal* in [2]. In Goal Oriented, *And/Or operationalization* link [11] are introduced to relate goal with the pre,

post and trigger condition in an operation which also can fall into this category. In addition, it is arguable that, *Requires* relationship described in [1] and *Effort* relationship covered in [17] could be considered similar to and can be classified as temporal relationship

- Two or more requirement should be implemented and satisfied at the same time. For example in Figure 5, Requirement 1 and 2 must be satisfied before requirement 3 is implemented and satisfied. This relationship is addressed in Feature Oriented as *Collateral* [7]. On the other hand in Aspect Oriented, *Pure interleaving* is proposed by Brito and Moreira that is equivalent [14].
- Two or more requirement should be synchronized sometime during their active period. It also means that two or more requirements should be implemented, satisfied and interact with each other at the same time. For example, a requirement saying “play a CD movie” for a DVD player only can function correctly if at the same time a television is functioned and there are connection between both electrical appliances. This kind of relationship is introduced as *synergetic* in [7] and *Full synchronization* in [14].

Requirement 1: A person shall registered as student Requirement 2: The system shall enable student to login Requirement 3: The system shall enable student register for one or more courses
---

**Figure 5: Example of Temporal Relationship Type**

### 3.4. Causality

Causality relationship is concerned with the cause, effects and consequences of the changing requirements. This relationship describes the change and impacts of one set of requirements have on another set of requirements and capture the history or version of a specific set of requirements. Furthermore, the characteristics relationships perhaps can be categorized under temporal relationship but we prefer to put it separately to specifically address the evolution of requirements as they change.

For example, in Figure 6, requirement 1 is related to requirement 2, 3 and 4 where requirement 1 is the initial requirement to be implemented; requirement 2 is a consequence of implementing requirement 1; but requirement 2 has changed to requirement 3 where requirement 4 is the effect of the changing process. In another aspect, Requirement 4 is the impact of implementing requirement 3. This relationship is addressed by Dahlstedt and Persson [1] where they name this relationship as *Change\_to*. Lee and Zhao [7] address the same category but by providing a detail discussion of the type of changes such as state change,

behavioral change and many others. In addition, in Feature Oriented approach, this type of relationship is referred to as *Impacts* [9] and *influences* [8].

Requirement 1: The system shall enable student to fill in the registration form. Requirement 2: The system shall generate the registration slip. Requirement 3: The system shall generate the registration slip only after authorization from registrar. Requirement 4: The system shall not generate the registration slip without authorization from registrar.
--

**Figure 6: Example of Causality Relationship Type**

### 3.5. Necessity

Necessity relationship is concerned with the fact that a specific set of requirements needs another set of requirements to be satisfied. Specific set of requirements might be dependent and constrain others in a specific situation or environment. This relationship is also concerned with some situation where a requirement is a pre-condition or pre-requisite for another requirement.

- The implementation of a set of requirements might be dependant on another set of requirements to function and accomplish a task. This kind of relationship is addressed by many researchers as *requires*, *necessity* or *usage* [1, 2, 6, 9, 11, 14, 17]. For example in Figure 7, requirement 1 could require requirement 2 to function. Requirement 3 and 4 are an example of how two requirements related to and need each other in accomplishing a particular task. This kind of relationship is represented by *bidirectional* relationship proposed as *And* in [2] and *bidirectional necessity* in [17]. In addition, *Task/Goal* and *Resource* dependency in [13] which represent how an actor dependent on another actor to achieve a goal and accomplish task can be categorized into this classification.
- A set of requirements is constrained by the capability of implementing another set of requirements. For example in Figure 7, Requirement 2, if not satisfied typically can cause Requirement 1 and 3 not to be satisfied and implemented. This relationship is addressed by several researchers [7, 8, 13]. Softgoal which can represent non-functional requirements (e.g. performance requirements) as proposed in [10, 12] can typically constrain the implementation of other requirements.

Requirement 1: The system shall enable student to register for courses online Requirement 2: Network connection is facilitated and functioned correctly Requirement 3: The system shall provide the forum and email services for all students. Requirement 4: The system shall bill students per minute when they use the forum and email services.
--

**Figure 7: Example of Necessity Relationship Types**

#### **4. Requirements Relationships in Software Testing Practices**

Testing is one way of ensuring that all requirements of the system have been met. Many studies show that ‘delayed testing’ leads to stressful and costly test and maintenance phases [20]. Thus, it is important to do validation and verification (testing) from the initial phases, specifically from requirements phase. This significant interrelationship between testing and requirements provides the possibility of performing testing activities at the early stages of software development. Requirements based testing has been addressed by previous researches (e.g. [21]) but it is rare to find work specifically addressing requirements relationships in the context of testing or test cases. Some researchers relate requirements and testing activities in the context of traceability (e.g.[22]) but not at the right level of granularity.

As testing will involve much effort and resources, there are studies that discuss the reduction and minimization of test suite or test cases. Most of the studies focus on the minimal test suite or test cases selection for regression testing [23, 24]. Some investigate the use of requirements relationships but only based on control and data dependency between requirements or components to reduce test suite for testing (e.g. [25]). In addition, Chen et al [26] propose an approach of test suite reduction based on requirements relation contraction by identifying and removing the redundant requirements using graph theory. However, Chen et al just focus on the redundancy between requirements and not explicitly discuss the types of requirements relations that may exist [26]. Chittimali and Harrold indicate that they have discovered situations in which test cases were used for multiple requirements [27].

There are related tools introduced in practice (e.g. Requisite Pro, DOORS). Most of the tools facilitate the traceability between requirements and test cases but not articulate the linkage process between different types of requirements. Other tools such as Quality Centre focuses on test management where the tools can link test cases, into the related requirements, test plan and test suite [28]. Tools such as DOORS and Caliber RM are requirements management tools where we can link different type of artifacts to requirements but need to combine several tools to facilitate the process for requirements management linkage to testing [29, 30].

#### **4.1. Structural Relationships and Testing**

Structural relationship which addresses parent-child relationship and how a group of requirements can be organized in structures is crucial for testing activities.

- As the first subtype of structural relationships, **Refinement** may help tester by telling which part of the hierarchy need to be tested first. The existence of this relationship requires the validation and verification activities be initiated with the parent as the abstract requirement. The validation activities then require the detailed requirements to be tested to distinguish and verify that the abstract requirement has been satisfied. For example, as illustrated in Figure 1, Requirement 2 and Requirement 3 need to be tested to ensure the operation of Requirements 1 has been correctly and completely accomplished. In order to test requirements with refinement relationship, tester has to traverse along the entire path in the hierarchy to ensure that all the related requirements have been validated. The refinement relationship knowledge that has been identified can contribute to the completeness of the coverage. Consequently, the knowledge can also be used as the basis for developing the test plan. As requirements can be traced to the related test cases, knowing the number of each requirement to be validated in sequence may provide information for organizing the test plan.
- **‘Is-a’** relationship demonstrates how child requirements inherit the characteristics of their parent requirement. This relationship knowledge may help tester in the reduction of test cases as the parent and child requirements might be grouped together in one test case. Moreover, the testing for a component related to the parent requirements can subsume the component testing related to the child requirements. For instance in Figure 2, if a payment using cash card has been made which means requirement 2 has been satisfied, it also means requirement 1 has been satisfied. If one of the requirements 1, 2 or 3 has been satisfied, we can consider that all the requirements have been satisfied. This relationship if identified and managed may help tester to minimize test cases where we can use the same test cases for all classes of requirements 1, 2 and 3.
- Finally, **Aggregation** or whole part relationship if identified may help tester to recognize which part of the system as the children or sub-module need other part and the main module as the whole to be tested together. The validation and verification activities of the children need a reference to the parent to be valid. This is important to ensure that the functionality of the group of requirements can be validated

accurately. For instance in Figure 3, the combination of requirements 1.1 and 1.2 depends on requirement 1 as a whole to complete the registration process. Thus, we need the whole and its parts to validate the completeness of the process. This relationship is important especially to the component based software development and may also be used to ensure the completeness of the coverage for testing.

#### 4.2. Necessity Relationships and Testing

Necessity relationship will help the tester by indicating which requirements need or constrained other requirements in order to be fully satisfied. This information may help the tester to determine which part of the system related to the requirement needs other requirements as the pre-requisite to ensure the functionality. The pre-requisite requirements should be validated before the related requirements. As test cases can be traced back to related requirements, necessity relationships may also tell the order of the related test cases. Hence, necessity relationship is also important for the completeness of the testing coverage and for the accuracy of the test plan.

#### 4.3 Temporal relationship and testing

Temporal relationship is concerned with actions in a specified temporal order. One instance of temporal relationship allows coordination of the requirements during sequential implementation. This relationship may inform the tester of how to validate the requirements according to the temporal manner of pre and post condition.

#### 4.4 Causality relationship and testing

This relationship is introduced to address the condition of changing requirements. Causality relationship if identified will provide information of which set of requirements has changed to another set of requirements and the effects of the changes. When a requirement has changed to another, the knowledge may help tester to identify which related test cases are obsolete and can be removed from the test suite. The change information and causality relationships may reveal several test cases that are no longer needed to be validated. Thus, causality relationship is important to be considered in testing particularly for the coverage and minimization of the test suite.

#### 4.5 Implementation Relationships and Testing

There is an instance of *implementation* relationship which is *similar* relationship that is significant in testing. This relationship will help tester by telling which requirements are similar. If several requirements

are similar to one another, only one of them needs to be implemented and tested. Hence, we may use or reuse the same test cases which can result in minimization of test case creation.

## 5. Conclusions

Requirements relationships knowledge describes how requirements are related to one another in many different ways. In this paper, we presented a preliminary classification of requirements relationships providing examples of each type. We have also compared and contrasted these relationships from major bodies of work to illustrate similarities and differences between them. An important issue to be addressed then is how software developers could use this classification in an effective manner in practice. Thus, in the second part of this paper, we addressed the use of requirements relationships in testing practices. We discussed how the classification of requirements relationships can be utilized to help testing activities.

We intend to extend our study to develop a comprehensive catalogue of requirements relationships not just from literature but by conducting field studies from practice. This is important, as there maybe related tacit knowledge about requirements relationships which can only be discovered from a closer look at practice. Then, we intend to formalize the classification and use the classification to improve software testing practices.

## 6. References

- [1] A. G. Dahlstedt and A. Persson, "Requirements Interdependencies: State of The Art and Future Challenges," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Germany: Springer-Verlag Berlin Heidelberg, 2005, pp. 95-116.
- [2] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt Och Dag, "An Industrial survey of requirements interdependencies in software product release planning," in *Proceedings of Fifth IEEE international symposium on Requirements Engineering*, Toronto, Canada, 2001, pp. 84-91.
- [3] A. M. Davis, "The Art of Requirements Triage," *IEEE Computer*, pp. 42-49, March 2003.
- [4] J. Kuusela and J. Savolainen, "Requirements Engineering for Product Families," in *Proceeding 22nd international conference on Software engineering*, Limerick, Ireland, 2000, pp. 61-69.
- [5] S. Robertson and J. Robertson, *Mastering the Requirements Process second edition*. Boston: Addison Wesley, 2006.



- [6] S. Buhne, Halmans, G. & Pohl, K., "Modeling Dependencies between variation points in use case diagrams," *Proceedings of the 9th international workshop on Requirement Engineering-foundation for software quality REFSQ'03*, pp. 59-70, 2003.
- [7] Y. Lee and W. Zhao, "A Feature Oriented Approach to Manage Domain Requirements Dependencies in software Product Lines," *Proceedings of the First Multi-Symposiums on Computer and Computational Sciences IMSCCS'06*, 2006.
- [8] W. Zhang, Mei, H. & Zhao, H., "A feature-oriented Approach to Modeling Requirements Dependencies," *Proceedings of the 2005 13th international Conference on Requirement Engineering RE'05*, 2005.
- [9] H. L. Ye, H., "Approach to modeling feature variability and dependencies in software product lines," *IEE Proceedings online*, vol. 152, pp. 101-109, June 2005 2005.
- [10] J. Clealand-Huang, R. Settini, O. B. Khadra, E. Berezanskaya, and S. Christina, "Goal-centric traceability for managing non-functional requirements," in *27th International Conference on Software Engineering (ICSE 2005)*, St. Louis, Missouri, USA, 2005, pp. 362-371.
- [11] A. V. Lamsweerde, "Goal Oriented Requirements Engineering: A Guided Tour," in *Proceedings RE'01, 5th international Symposium on Requirement Engineering* Toronto, Canada, 2001, pp. 249-263.
- [12] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering "in *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, Anapolis, USA, 1997, pp. 226-235.
- [13] J. Araujo, E. Baniassad, P. Clements, A. Moreira, A. Rashid, and B. Tekinerdogan, "Early Aspects: The Current Landscape," Lancaster University, Technical Report COMP-001-2005, February 2005 2005.
- [14] I. S. Brito and A. Moreira, "Advanced Separation of Concerns for Requirements Engineering," *Journal of Software Engineering and Databases* vol. 8, 2003.
- [15] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements Interaction Management," *ACM Computing Surveys*, vol. 35, pp. 132-190, June 2003 2003.
- [16] L. Karlsson, A. Dahlstedt, B. Regnell, J. N. o. Dag, and A. Perrson, "Requirements Engineering challenges in market driven software development-An interview study with practitioners," *Information and Software Technology*, vol. 49, pp. 588-604, 2007.
- [17] A. M. Davis, *Just Enough Requirements Management : Where Software Development Meets Marketing*. New York: Dorset House Publishing, 2005.
- [18] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. USA: Addison-Wesley, 1999.
- [19] P. Gray, *Logic, Algebra and Databases*. London: Ellis Horwood Limited, 1984.
- [20] C. Denger and T. Olsson, "Quality Assurance in Requirements Engineering," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Germany: Springer-Verlag Berlin Heidelberg, 2005, pp. 163-185.
- [21] C. Nebut, F. Fleurey, Y. L. Traon, and J. Jezequel, "Automatic Test Generation: A Use Case Driven Approach," *IEEE Transactions on Software Engineering*, vol. 32, pp. 140-155, March 2006.
- [22] M. Lormans and A. V. Deursen, "Reconstructing Requirements Coverage Views from Design and Test using Traceability Recovery via LSI," in *In proceedings of TEFSE'05, International Workshop on Traceability in Emerging Forms of Software Engineering*, Long Beach, California, USA, 2005, pp. 37-42.
- [23] J. Zheng, "In regression testing selection when source code is not available," in *ASE'05*, Long Beach, Claifornia, USA, 2005, pp. 452-455.
- [24] G. Rothermel and M. J. Harrold, "Analyzing Regression Test selection Techniques," *IEEE Transactions on Software Engineering*, vol. 22, pp. 529-551, 1996.
- [25] S. Jungmayr, "Identifying Test-Critical Dependencies," in *Proceedings of the International Conference on Software Maintenance (ICSM'02)*, 2002.
- [26] Z. Chen, B. Xu, X. Zhang, and C. Nie, "A Novel approach for Test Suite Reduction Based on Requirement Relation Contraction," in *SAC'08 Fortaleza, Ceara, Brazil*, ACM, 2008.
- [27] P. K. Chittimalli and M. J. Harrold, "Regression Test Selection on System Requirements," in *ISEC'08 Hyderabad, India*: ACM, 2008.
- [28] I. Checkpoint\_Technology, "Hp Quality Center," in *Hp Test Director for quality Centre*. vol. 2008 Florida: hp Mercury Certified Training Partner, 2008.
- [29] T. Jones, "Technology audit," in *Requirements Management Solutions: Telelogic*: Butler Group, 2007, pp. 1-10.
- [30] Borland, "Borland CarliberRM:Enterprise Software Requirements Management System ". vol. 2008 USA: Borland, 2008.

# Towards the selection of the most suitable elicitation technique through a defined requirements elicitation process

Marcelo Werneck Barbosa<sup>1</sup>, Glívia Angélica Rodrigues Barbosa<sup>1,2</sup>

<sup>1</sup>*Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais (PUC Minas) – Belo Horizonte – MG – Brasil*

<sup>2</sup>*Newcom Brasil – Belo Horizonte – MG - Brasil*  
{mwerneck}@pucminas.br,{glivia.barbosa}@newcombrasil.com.br

## Abstract

*In an attempt to ensure that the relevant system requirements are correctly and completely elicited, a set of techniques can be applied aiming at helping analysts and users identify and define these requirements. However, the elicitation is not simply the application of a technique but also the cooperation among analysts and clients. This paper presents a requirements elicitation process that focuses on aiding analysts selecting the best elicitation technique to be used in a particular project. The mechanism used to select the most suitable elicitation technique has also been confronted with recent studies in this field.*

## 1. Introduction

One critical step in Requirements Engineering is elicitation, a complex phase of requirements definition since it is the foundation to all upcoming phases [2]. It demands an iterative process, which may be executed collaboratively and it involves the use of techniques [5].

Elicitation techniques contribute to software development, but essential problems related to eliciting requirements are a challenge that is yet to be overcome.

Selecting and applying an elicitation technique is not a trivial task, however, narrowing the elicitation phase to the use of techniques only does not guarantee that identified requirements fulfill clients' needs [2].

It has been observed, however, that the way these techniques have been applied has not yet solved the problems in elicitation. Requirements elicitation is generally performed using an elicitation methodology or a series of techniques. Several techniques exist, all with the common goal to assist analysts in understanding users' needs. Although some analysts think that just one technique is applicable to all situations, however, one technique cannot possibly be sufficient for all conditions [6]. Information is still captured, in most cases, using interviews only, although there is clear evidence that traditional interviews are not always the best option [3]. Despite the critical need for eliciting the right

requirements, little research has been focused on identifying the most adequate elicitation techniques [4].

Considering requirements elicitation a critical phase, this work proposes a requirements elicitation process focused on selecting the most suitable elicitation technique based on project's characteristics. This process provides more interaction among analysts and users and also aids in selecting the technique to be used to yield requirements closer to user's needs. The use of a defined process has shown several benefits, however, the selection of a technique demands further research. Since technique selection is a tricky subject, the mechanism used to select the most suitable elicitation technique in the process has also been confronted with recent studies in this field. It has been observed that more research is needed even considering that a vast study on the literature has recently been presented [3] and [4].

This work is organized as follows. Section 2 describes related work. Section 3 describes the elicitation process proposed while Section 4 presents the results achieved. Section 5 concludes the paper and presents possibilities of future work.

## 2. Related work

In [2], an analysis is performed in order to compare and present differences and similarities among elicitation techniques. The authors proposed a set of parameters to assess and classify some of the studied techniques.

Another study related to the subject was described in [1]. The authors present some parameters related to projects and techniques that must be analyzed when selecting the elicitation technique.

In [5], a requirements elicitation collaborative process is described. The authors present a process and a supporting tool. Through a case study, it has been concluded that the collaborative process improved the communication among stakeholders. No analysis of technique selection has been performed.

The work [4] presents some recommendations in which elicitation techniques are useful. They are based on a systematic review with several empirical studies.

In [3], framework is presented to support developer decision-making on which the best elicitation techniques for the project at hand are. The framework identifies which elicitation techniques respond better to certain project features. A set of project attributes influencing technique effectiveness was determined. Finally, all information gathered was compiled in a framework that matches elicitation techniques to project attributes.

This work stands out since it defines a detailed requirements elicitation process with support to better selecting the elicitation technique as well as providing greater integration among stakeholders. The mechanism used to select the most suitable elicitation technique has also been confronted with recent studies [3] and [4].

### 3. The Requirements Elicitation Process

The requirements elicitation process proposed comprises five activities, described as follows:

1. The first activity, **“Identifying the context of the project”**, is aimed at contextualizing the requirements analyst on the project being developed. In a meeting with the project manager, the analyst obtains information related to project scope, assumptions, constraints and the client domain before having contact with users. At the end of this activity, a glossary is elaborated with the main definitions identified.

2. The second activity, **“Performing initial project presentation”** comprises performing a meeting with all stakeholders to highlight the importance of user collaboration in the elicitation. The analyst also collects information on roles and responsibilities in the elicitation. A Responsibility Matrix is elaborated.

3. The following activity, **“Selecting the requirements elicitation technique”**, focuses on aiding the Analyst choosing the most suitable elicitation technique according to the project’s characteristics, the organization and the project team.

The process does not restrict the number of techniques to be chosen from. However, in the experiments, five were considered. These techniques are a subset of the techniques analyzed in [1] and [2]. Techniques that could be easily applied in the organizations in which the experiments were executed were selected: Brainstorming, Interview, JAD, Questionnaires and Prototypes.

The techniques are selected according to some parameters. Each technique received a score for each parameter, indicating to which degree the technique is successful in achieving what is being evaluated. The score given was based on [1] and [2].

In order to support the elicitation technique selection, a decision matrix is used. It consists of weighting possible solutions (the techniques) against parameters.

Each technique is assigned a score to each parameter according to [1] and [2]. Each technique has been

assigned classifications from “Low” to “High” for each parameter. Each classification was mapped to a value from 1 to 3. Since the goal is to assign the highest score to the most suitable technique, values were assigned depending on the main goal – minimize or maximize the effect of the parameter. In this way:

- Parameters (characteristics) that should be maximized, like quality, were assigned scores from 1 to 3, corresponding from “Low” to “High”.
- Parameters that should be minimized, like time, were assigned scores from 1 to 3, representing values from “High” to “Low”.

The analyst defines which parameters should be prioritized based on projects’ characteristics. This prioritization is performed by assigning a degree from 0 to 5, indicating how important that parameter is in the project. For example, when analyzing “quality”, the analyst should ask herself to what degree the project demands a technique focused on the quality of the elicited requirements. Other parameters analyzed were related to cost, validation, training needs, etc ...

By summing up the outcome of the multiplication of the score assigned to the technique and the weight the analyst has given to that parameter, a final value is achieved. The technique that has achieved the highest value is possibly the most suitable.

4. The activity **“Applying the requirements elicitation technique”** aims at applying the selected technique to elicit the project’s requirements. As a result, a document called “Elicited Requirements” is elaborated.

5. The last activity is **“Elaborating the requirements list”**. The requirements lists should contain a description of the requirements and acceptance criteria for each one. The Requirements list is inspected through a verification process not included in this elicitation process.

### 4. Experiments and Results Achieved

In order to validate the process proposed, it has been performed in 4 projects in 3 different companies. For each project, the elicitation was carried out twice: one using the process and other using the company’s process or no defined process if the company did not have any.

In order to compare both elicitations’ results, data on time, cost, quality and context were collected. Quality and context were measured based on a verification checklist filled in by the client. This checklist comprises verification items that analyze if the elicited requirements are complete, precise, consistent and clear.

This checklist is a sheet in which elicited requirements are placed in lines while verification items are placed in columns. In the intersection of each cell, the client registers her analysis: if the requirement is compliant, partially compliant, not compliant or if she does not know how to assess the requirement. In order to easily

assess the verification process, each possible classification received a numeric value. Compliant items scored 1; partially compliant items scored 0.5 while not compliant items scored 0.

The selection of the most suitable technique is performed based on the degree of each parameter being assessed. In order to evaluate if the selected technique really fulfilled the needs of the analyst, each verification item in the checklist was associated with one parameter (quality, context, ...) being analyzed. This relationship indicates that if the verification item is completely compliant, the corresponding parameter will be optimized.

Since each elicitation (with and without the process) may identify a different number of requirements, results analysis is performed based on the total number of unique requirements identified in both elicitations. This number has been called "Requirements Universe".

A different analyst conducted each elicitation. The analysts chosen for each execution had similar skills, education, background and experience with elicitation.

The results achieved with these experiments are displayed in Table 1. It can be observed that in all cases the elicited requirements through the proposed elicitation process presented a higher level of quality, according to the client's verification. This result is related to the fact that the requirements analyst better explores the context of the system and the business domain of the client.

Although in some case studies, the same elicitation technique was used in both executions (with and without the process), the elicited requirements using the process were assessed by the client as having a higher degree of quality and a better context understanding.

It can also be observed in Table 1 that in some case studies, elicitation took longer when using the process. This small difference, however, is seen as investment to better elicit requirements. In these projects, according to the parameters defined by the analyst, time was not a constraint. Moreover, requirements elicited with the process were better evaluated by the client.

**Table 1. Summarized Results**

Collected Data and Parameters Evaluated	Case Study 1 Ticket Reservation		Case Study 2 E-commerce		Case Study 3 Bug Tracking		Case Study 4 Shop Management	
	With process	Without process	With process	Without process	With process	Without process	With process	Without process
Requirements Universe	21		23		24		35	
Technique	Prototype	Interview	Interview	Interview	Interview	Interview	Brainstorming	Interview
Time (hours)	6	8	2,5	2	2,5	1	3,5	1
# of Elicited Requirements	21	6	23	15	24	11	35	11
Percentage (Universe)*	100%	29%	100%	65%	100%	46%	100%	31%
Quality**	98%	100%	100%	74%	98%	75%	95%	51%
Weighted Quality***	98%	29%	100%	48%	98%	34%	95%	16%
Context****	100%	50%	100%	50%	100%	25%	100%	25%
*	Represents the number of elicited requirements identified in each elicitation in relation to the number of total requirements identified by the client in both executions of the case study (with and without the process)							
**	Represents the conformity degree of the elicited requirements for each elicitation in relation to the items in the verification checklist related to the quality parameter (according to the verification carried out by the client)							
***	Represents the conformity degree of the elicited requirements for each elicitation in relation to the items in the verification checklist related to quality (based on the verification done by the client) weighted by Requirements Universe.							
****	Represents the conformity degree of the elicited requirements for each elicitation in relation to the items in the verification checklist related to the context parameter (according to the verification carried out by the client)							

#### 4.1. Validating the selection of the elicitation technique

It has been observed with the reported results and the application of the selection technique that differences among elicitation techniques could be better explored and highlighted in the decision matrix. The works this process has been based on ([1] and [2]) define parameters with the same value (weight) for

several techniques. If higher ranges had been used, differences could have been deeply explored.

Aiming at validating if this process has correctly chosen the most suitable techniques, these results were confronted to findings of recent published work ([3] and [4]). These papers have thoroughly analyzed several elicitation techniques against more parameters. This analysis has been conducted through a deep review of the literature on elicitation techniques.

In [3], a framework that aids the selection of the best elicitation technique was proposed. Information found in the literature about when and where it is appropriate to use each elicitation technique was analyzed. A set of project attributes influencing technique effectiveness was determined. Finally, all information gathered was compiled in a framework that matches elicitation techniques to project attributes. The different technique adequacy levels were classified into adequate, indifferent or having a low adequacy level.

The decision matrix used in our experiments was updated to consider exactly the same criteria and values defined in [3]. The same analysts who conducted each experiment described previously filled in the updated decision matrix for the same projects. The goal of this comparison was to analyze if the framework would

point out different techniques and if these could be considered more suitable to each scenario.

Surprisingly, Questionnaire was selected in all four executions considering the updated Decision Matrix, even in projects in which few informants were available. Questionnaires are usually used to collect information from several users at the same time [7].

Analyzing the framework proposed [3], it is possible to see that the Questionnaire technique has been classified as adequate for the vast majority of attribute values analyzed (86.36%). This suggests that no other technique would probably be selected in most cases.

Requirements have not been re-elicited using Questionnaire to verify if there could be even better results. It is planned as future work. Table 2 displays the techniques selected by the new process executions.

**Table 2. Process Selection Technique X Framework Selection Technique**

Technique (This Process)	Case Study 1 Ticket Reservation		Case Study 2 E-commerce		Case Study 3 Bug Tracking		Case Study 4 Shop Management	
	With process	Without process	With process	Without process	With process	Without process	With process	Without process
	Prototype	Interview	Interview	Interview	Interview	Interview	Brainstorming	Interview
<i>Technique (Framework [3])</i>	<i>Questionnaire</i>		<i>Questionnaire</i>		<i>Questionnaire</i>		<i>Questionnaire</i>	

## 5. Conclusions and future work

In this work, a collaborative process that supports analysts on selecting the most suitable elicitation technique, based on project attributes was proposed.

The requirements elicited through the proposed process presented better and improved characteristics compared to the ones obtained through other methodologies. This is due to the prioritization of parameters relevant to the project while selecting the technique as well as allowing stakeholders and analysts to have better collaboration and interaction.

In the cases in which the proposed process was executed, the techniques suggested fulfilled the criteria established by analysts and the elicited requirements were successfully verified by clients. This suggests that the thorough understanding of the project context, the establishment of criteria to select the elicitation technique and the collaboration among stakeholders contribute to an elicitation closer to the real needs.

The case studies have not comprised requirements development. Also, analysts' profile in each elicitation may influence the outcome, so extra care has been taken to choose analysts with very close characteristics.

The results of the process have been confronted with a recently published framework [3]. Even though such work consists of a vast review of the literature on elicitation techniques, its use might need to be adjusted by the execution of different experimental studies.

## 6. References

- [1] Batista, E. A.; Carvalho, A. M. B. R. (2003) "Uma Taxonomia Facetada para Técnicas de Elicitação de Requisitos". Workshop em Engenharia de Requisitos 2003.
- [2] Belgamo, A. e Martins, L. E. G (2000). "Estudo Comparativo sobre as Técnicas de Elicitação de Requisitos do Software". In: XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba – Paraná.
- [3] Carrizo, D.; Dieste, O. e Juristo, N. (2008) "Study of Elicitation Techniques Adequacy". In Workshop on Engenharia de Requisitos (WER 2008), Barcelona, Espanha.
- [4] Dieste, O.; Lopez, M. e Ramos, F. (2008) "Updating Systematic Review about Selection of Software Requirements Elicitation Techniques". In Workshop on Engenharia de Requisitos (WER 2008), Barcelona, Espanha.
- [5] Freitas, D. P.; Borges, M. R. S; Araújo, R. M. (2007) "Colaboração e Negociação na Elicitação de Requisitos" In: X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS 07), Isla de Margarita
- [6] Hickey, A. M. e Davis, A. M. (2002) "Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes". In: Proceedings of Hawaii International Conference on System Sciences
- [7] Lauesen, S. (2002) "Software Requirements Styles and Techniques". Elicitation. England: A Personal Education Limited, 2002. Cap.8 p.331-372.

# A Requirement Traceability Refinement Method Based on Relevance Feedback

Lingjun Kong<sup>1,2</sup>, Juan Li<sup>1</sup>, Yin Li<sup>1,2</sup>, Ye Yang<sup>1</sup>, Qing Wang<sup>1</sup>  
<sup>1</sup>Laboratory for Internet Software Technologies, Institute of Software  
<sup>2</sup>Graduate University of Chinese Academy of Sciences  
{konglingjun, lijuan, liyin, ye, wq}@itechs.iscas.ac.cn

## Abstract

*In this paper, we conduct a study of using relevance feedback-based Information Retrieval (IR) methods to refine Requirement Traceability (RT) from requirement to code. We compare two representative feedback methods: Mixture Model (MM) in language model and Standard Rochio method (SR) in vector-space model. In order to assure the fairness of comparison, we also make modification for both of the methods. Initial experiment results on a real project data set show that 1) few iterations of feedback result in significant increases both in precision and recall; 2) feedback methods in language model are generally more stable than methods in vector-space model in improving precision, but the latter is more effective and can get better precision; 3) negative feedback information plays an important role in refining requirement traceability.*

## 1. Introduction

Despite the existence and increasing adoption of Computer-Aided Software Engineering (CASE) tools, there are still many software projects in which no Requirement Traceability (RT) exists. One possible reason is that RT is generated manually in most of existing tools assisting software development, such as DOORS [18]. This often leads to problems that RT is hard to maintain, error-prone and overrunning cost [1-4]. To alleviate these problems, Dynamic Requirement Traceability (DRT) adopts Information Retrieval (IR) technologies to help analyst automate RT [1-5]. The practice in DRT has shown that it is a better way to establish and maintain RT. However, DRT suffers from the precision problem [4].

Many methods have been proposed to deal with precision problem of RT, such as Latent Semantic Indexing (LSI) [14], key-phrases, simple thesaurus [2] and so on. In IR, relevance feedback, as an important

research topic for well over decades, has been an effective technique to improve the performance of the retrieval [12]. The main process of feedback can be described as: first do an initial retrieval and next update the query based on user's evaluation on the retrieved documents, and then do retrieval again with the new query which will hopefully have better retrieval performance [7, 9]. The evaluation mainly consists of user's effort to verify the retrieved documents relevant (positive feedback information) or irrelevant (negative feedback information) to user's query. Generally, there are many different relevance feedback methods in IR and Standard Rochio method (SR) in vector-space model [16] and Mixture Model (MM) in language model [7] are two representative feedback methods. The idea of SR is to update a query with both relevant and irrelevant documents. While MM only uses the relevant documents to update the query. Taking RT establishment as an IR problem, it is natural to use analyst's feedback to refine RT. Hayes has used SR to process analyst's feedback [3]. The experiment result of Hayes has shown that relevance feedback method has a good performance in improving precision of RT [3].

We conduct a further study of using relevance feedback-based IR methods to refine RT from requirement to code in this paper. Vector-space model is a common model in generating RT. We adopt a new retrieval model: language model to build traceability. This IR model is not used to generate RT in the work prior to ours. We use MM and SR to incorporate analyst's feedback respectively. We also compare both of feedback methods. In order to make a fair comparison, we have made modification for MM feedback method. Since MM naturally does not support negative feedback but SR does, extension has been made to make MM can integrate negative feedback which is called extended Mixture Model (eMM) in this paper. Experiments are conducted on a

real project data set to evaluate the effectiveness of feedback-based IR methods for RT.

The rest of this paper is organized as follows. In section 2, we review the related work. In section 3, we propose a RT establishment and refinement method. Experiment is analyzed and discussed in section 4. We conclude this paper and discuss our future work in section 5.

## 2. Related work

There are two areas of related work: first is IR methods adopted in this paper and second is methods proposed to refine RT.

1) Two IR models: Language Model (LM) [6, 7] and Vector Space Model (VSM) [9-11] are used to generate the requirement traces from requirement to code. The relative simplicity and effectiveness of LM, together with the fact that it leverages statistical methods that have been developed in many areas, make it an attractive way to develop new text retrieval methodology [7]. This is a major reason why we use LM in this paper. VSM is a commonly used model in generating RT [1-5].

2) Many methods have been proposed to deal with precision problem of RT. Andrian [13] adopted Latent Semantic Indexing (LSI) to compute the similarity scores between code and system documentation. Considering the work products' characteristics, J. C. Huang et al. introduced strategies for incorporating supporting information into a probabilistic retrieval algorithm to improve the performance of RT [4]. Hayes proposed several enhancement methods, TF-IDF (term frequency-inverse document frequency) + key-phrases and TF-IDF+Thesaurus with relevance feedback, to refine the requirement tracing [2, 3].

Our work is similar to Hayes' work in using analyst's feedback [3]. The major difference is that we adopt the new retrieval model: LM and we adopt many new LM based technologies such as relevance feedback model to improve the precision of RT. Moreover, the comparisons of feedback's performance between LM and VSM are reported in our paper.

## 3. Requirement traceability establishment and refinement method

In order to evaluate the effectiveness of analyst feedback-based IR methods for RT, we propose a RT establishment and refinement method to build traces from requirement to code automatically and adopt analyst's feedback to improve the precision.

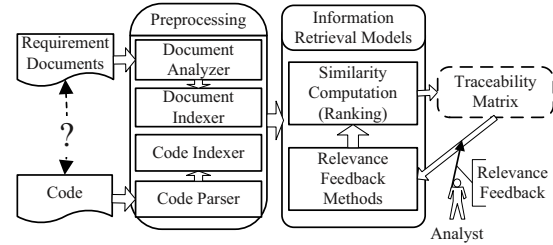


Figure 1. Requirement traceability establishment and refinement method

As shown in Figure 1, the establishment and refinement method can be divided into three steps: 1) preprocess requirements and code data, 2) establish the initial requirement traces using IR models and 3) incorporate analyst's feedback to refine traceability. The rest of this section describes these 3 steps in detail.

### 3.1 Data preprocessing

The main purpose of this phase is to preprocess the documents and code data, including text parsing, information extracting and indexing and so on. The inputs are requirement documents and code. The outputs are indexed texts.

#### (1) Preprocess requirement documentation

Requirement in this paper represents the software requirement which is described in a common tree structured word document. Every node in the tree is a requirement record. We use Document Analyzer to parse and split requirement documentation into many subdocuments with different granularity (set by analyst). Every output subdocument contains a requirement record.

#### (2) Preprocess code

Code Parser is used to traverse and parse all code files as well as extract information of classes, such as class name, method name and attribute name. Each output file contains extracted information of one class.

#### (3) Index documents

After preprocessing requirement and code, we index the outputs of Document Analyzer and Code Parser respectively using Document Indexer and Code Indexer. Both of Indexers are built based on Lucene [14], including text parsing, tokenizing and indexing.

### 3.2 Establishment of initial requirement traces

The initial requirement traces indicate initial retrieval results prior to any analyst's feedback. Establishing initial requirement traces is a retrieval process of computing the relevance or similarity between requirements and code which is mainly dealt

with by Similarity Computation (SC). The process is similar to search the internet using, for example, Google. The requirement and code are expressed as query and document respectively [5]. We adopt KL-Divergence [7] method to compute the similarity in LM. This method is one of the most effective retrieval models in LM. In order to avoid the sparse data problem [6], we have chosen the Jelinek-Mercer smoothing method which outperforms others for long queries (the query is generally long in generating traces) [8]. As to VSM, we use a famous metric TF-IDF to compute the weight of the index term and use the cosine similarity [9] between the requirement and code to measure the similarity [9-11]. More detail about similarity computation can be seen in [14]. The candidate traces are ranked by the similarity value in descending order and top N (set by analyst) traces are represented to the analyst which is shown as “Traceability Matrix” in Figure 1.

### 3.3 Refinement of Requirement traceability

As presented before, two representative methods: MM in LM and SR in VSM are used to improve the precision. We also make modifications for both of them so as to do solid and fair comparisons. Table 1 lists 4 feedback methods.

Table 1. Feedback methods

IR Models	Feedback Methods	Simple Description
VSM	SR	Standard Rochio method in VSM
	nSR	modification of SR (omitting negative feedback in SR)
LM	MM	Mixture Model in LM
	eMM	extension of MM (adding negative feedback to MM)

#### (1) SR and nSR

The SR method refers to:

$$q' = \alpha q + \frac{\beta}{|D_r|} \sum_{d_j \in D_r} d_j - \frac{\gamma}{|D_n|} \sum_{d_j \in D_n} d_j$$

where  $q'$ ,  $q$  denote new query and original query.

$D_r$ ,  $D_n$  represent relevant document (positive feedback information) and non-relevant document (negative feedback information) respectively.  $\alpha, \beta, \gamma$  are constant weights. In this paper,  $\alpha, \beta, \gamma$  are set to 1.0, 0.75 and 0.25 respectively which is considered to have a good performance [10]. When  $\gamma$  is equal to 0, SR method omits negative

feedback information. We call this nSR (no negative feedback information in SR) in this paper.

#### (2) MM and eMM

MM is a classic feedback method in LM which is presented in detail in [7].

In order to incorporate the negative feedback information, we make a simple extension to MM:

$$q' = \alpha q + \beta \cdot \theta_r - \gamma \cdot \theta_n$$

where  $q'$ ,  $q$  denote new query and original query.

$\theta_r$ ,  $\theta_n$  indicate probabilistic models generating relevant and non-relevant feedback documents. Both probabilistic models are generated with MM.  $\alpha, \beta, \gamma$  are constant weights and assigned to 1.0, 0.5 and 0.5 in this paper. This preference performs best in our experiment. We call the extended Mixture Model eMM.

This extension has the similar idea with “SingleQuery” method introduced in [17]. And this method is considered to be a feasible and effective way to incorporate the negative feedback information [17].

In section 4.3, we would compare and analyze the performance of SR and eMM in improving precision. SR and nSR, MM and eMM are compared respectively to show the importance of negative feedback in refining RT.

## 4. Experiment and Evaluation

### 4.1 Objective and subject of experiments

Table 2. Project context

Project Characteristics	Description
Project type	Web based application
Development process	Iterative
Development tool	Java/applet/struts/jsp/ajax, mysql, tomcat4
Team size	5 members
Developer skill	Experience
Project duration	30 weeks
Project scale	43 use cases, 70 KLOC, 468 classes
Logical module number	7
Deployment package	pmreq.jar, pmapplet.jar, pmwss.jar

**Objective of Experiments:** In order to assess the effectiveness of feedback method, experiments are conducted on a real project data set and the objectives of our experiment are to answer the following 3 questions:

(1) Is relevant feedback method effective to refine RT?



(2) If feedback method's performance varies with the increase of number of iterations of feedback?

(3) Is there any performance difference between eMM and SR?

**Subject of Experiments:** We chose a real project which was developing a requirement management system in a Chinese software company. This system aims to provide seamless connection between requirement and development, which incorporates with another in-house software project management product. The project context is listed in Table 2. Requirements of the project are described in Chinese in a MS Word document.

#### 4.2 Steps of experiments

The experiments were conducted in the following steps:

(1) Establish correct traces. We invited experts to build traces between requirements and code which are used to do feedback methods' performance evaluation. Moreover, we evaluated the performance of feedback methods by comparing result of experiment using feedback methods with the correct traces built by experts.

(2) Preprocess requirement documentation and code. We first translated the Chinese requirement document into English manually. Secondly, the requirement document was split into 43 subdocuments. Every subdocument contained a requirement. Code was split into 468 class files. Generally, class name is a combination of several words (or abbreviation of word). Thirdly, we split class name into several words according to coding standards. It's the same with attribute name and method name.

(3) Establish the initial traces using LM and VSM respectively.

(4) Incorporate analyst's feedback. We used 4 feedback methods to improve the precision of traces respectively. The feedback was performed by a feedback simulator (similar to Hayes in [3]), i.e., the

feedback provided by the simulator was always correct [3]. We simulated 8 iterations of feedback on 4 different numbers of top documents by this way. Moreover, after each feedback, we first applied the feedback methods to reformulate query and next did retrieval again with the reformulated query, then recorded the output for further evaluation, and finally presented the results to analyst. This process continued until 8 iterations had been completed or all true links had been found.

#### 4.3 Results analysis and evaluation

In order to answer the three questions proposed in section 4.1, we analyzed the experiment results. We use recall and precision [5] as metrics to evaluate result. Recall measures the number of correct traces retrieved over the total number of correct traces, and precision measures the number of correct traces retrieved over the total number of retrieved traces.

Table 3 shows the experiment results. The first column shows the percentage of documents retained for each query (percent of top documents in the ranked list). We simulate the analyst's feedback on these retained documents. The third column is initial retrieved results with LM and VSM. The table also shows the precision and recall with different numbers of iterations of feedback. The last column is the rate of growth of precision and recall using feedback.

##### Question 1: Is relevant feedback method effective to refine RT?

According to the last column of Table 3, we got that in comparison with the initial evaluation, the average increase in recall of eMM for all thresholds was over 18%, it was 12.8% for precision. For SR, recall increased by 14.5% on average and precision increased by 10.7%.

As shown in Table 3, both feedback methods allowed us to find over 80% of all correct traces with a precision of 37% (see values in bold). Both of the feedback methods perform best when threshold is 0.15.

Table 3. Experiment results

Thresho ld	IR methods	Initial Evaluation (avg. Prec./Recall)	Evaluation with 8 Iterations of Feedback (avg. Prec./Recall)						% delta in Prec./Recall (all are increase)
			1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	6 <sup>th</sup>	7 <sup>th</sup>	8 <sup>th</sup>	
0.05	LM (eMM)	46.7/33.2	59/42	62.4/44.4	65.2/46.4	67/47.8	66.7/47.5	67.7/48	21/14.8
	VSM (SR)	50/35.6	57.6/41	60.9/43.3	63.8/45.4	70.4/50.1	68/48.5	70/49.8	20/14.2
0.1	LM (eMM)	36.2/51.5	41.7/59.3	45.4/64.7	45.7/65	49.5/70.5	49.7/70.8	49.7/70.8	13.5/19.3
	VSM (SR)	39.8/56.6	45.7/65	46.4/66.1	48.6/69.1	52.3/74.6	52.3/74.6	51.9/73.9	12.1/17.3
0.15	LM (eMM)	27.6/58.9	31.1/66.4	33.7/71.9	35.2/75.2	37.3/79.7	37.6/80.3	<b>37.6/80.3</b>	10/21.4
	VSM (SR)	32.1/68.5	35.1/74.9	36.7/78.3	36.9/78.9	<b>38.2/81.7</b>	37.9/81	<b>38.2/81.7</b>	6.1/13.2
0.2	LM (eMM)	23.2/66.1	26.4/75.2	27.9/79.7	28.8/82	30/85.4	30/85.4	30/85.4	6.8/19.3
	VSM (SR)	25.8/73.6	27.7/78.9	29/82.7	29.2/83	30.1/86	29.5/84	30.1/86	4.3/12.4

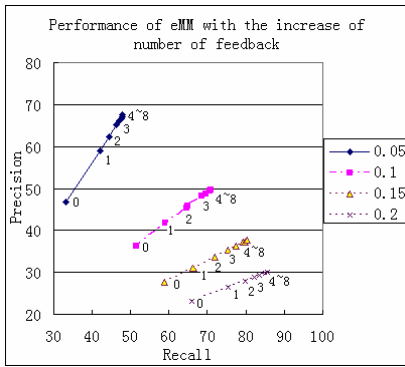
According to the evaluation standard introduced by Hayes in [3] (see Table 4), we can see that the result achieved “Good” combinations of precision and recall.

**Table 4.** Standards from Hayes

Measure	Acceptable	Good	Perfect
Recall	60%~69%	70%~79%	80%~100%
Precision	20%~29%	30%~49%	50%~100%

**Question 2: If feedback method’s performance varies with the increase of number of iterations of feedback?**

Figure 2 shows the performance of eMM with the increase number of iterations of feedback. Every line corresponds to a threshold in Table 3. There are 9 points in every line. Point 0 presents initial retrieval result and points 1 through 8 correspond to 8 iterations of feedback.



**Figure 2.** Performance of LM(eMM) with the increase number of iterations of feedback

As shown in Figure 2, first 2 iterations of feedback bring great precision improvement, while iterations from 3 to 8 have small growth. SR has the similar results (see Table 3). That is to say, few iterations of feedback result in significant increases both in precision and recall. This is a good result for analysts to use feedback to refine RT in real project. Because in practice, analysts may not be willing to spend time to improve the precision of traces by providing many iterations of feedback.

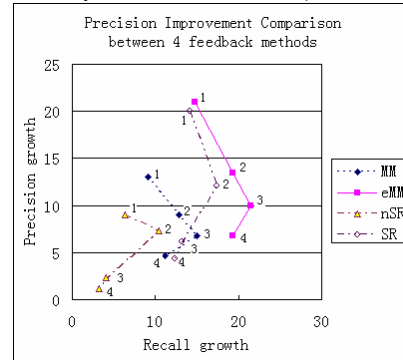
**Question 3: Is there any performance difference between eMM and SR?**

As shown in table 3, we could always get the highest precision and recall in the last feedback using eMM. However, the highest recall was not always found in the last feedback with SR. For example, for SR (threshold equals to 0.1), iteration 7 had a recall of 74.6% (it dropped to 73.9% in the last feedback). So eMM is more robust than SR in improving precision. However, we could get better precision with SR. For

example, for every threshold (except 0.05) and every feedback, SR had better precision and recall than eMM.

In the experiment, we also found that the negative feedback played a rather important role in adopting feedback to refine RT.

Figure 3 shows the average precision improvement in project data set using 4 feedback methods. As presented previously, eMM is extended to add negative feedback information to MM, while nSR is a transformation of SR by omitting the negative feedback information. Every point in the line of the Figure 3 corresponds to a threshold (see Table 3).



**Figure 3.** Precision growth comparison using 4 different feedback methods

As shown in Figure 3, without negative feedback information, performance of MM was considerably worse than eMM. For example, for all thresholds, the average precision growth using eMM was 5% higher than using MM and recall exceeded 7%. A similar observation was made for SR and nSR.

The possible explanation for this result is that the retrieval performance of IR methods is not very well, i.e. many traces retrieved in the top of the ranked list are irrelevant ones. Recalling earlier presentation, the class name, attribute name and method name are combinations of several words (or abbreviation of word). In fact, the abbreviation is more common. For example, a class named “ReqSearchMgrImpl” is a java class used to do operation to requirement in the project, such as deleting a requirement, getting requirement’s state and type. In our experiment, the name would be split into 4 words: “req”, “search”, “mgr” and “impl”. “req”, “mgr” and “impl” are the abbreviation of “requirement”, “management” and “implementation” respectively. Obviously, those abbreviations would not appear in the requirement documents which make the word in requirement documents fails to match the split words in class name. Therefore many irrelevant traces (negative feedback information) would appear in the top of the ranked list. It would be less effective if we do not use this negative information.

The finding that negative feedback is important for RT has an inspiration for building requirement traces in practice: it is rather meaningful for analyst to present irrelevant trace information to the system.

#### 4.4 Threats to validity

Two major threats to validity of our method are:

1) Ideal simulation. In this paper, we have simulated the ideal analyst feedback, i.e., the feedback provided by the simulator was always correct. However, the analysts may not be able to make a black or white judgment whether a trace is relevant or irrelevant in practice. At the same time, the judgment is a process of subjective evaluation and analysts may have different opinion on whether one trace is relevant or not. This may affect the performance of feedback.

2) Data size. It is insufficient to prove the efficiency of a method just with a data set. Currently, feedback method was applied in just one project. The precision of establishing requirement traces and performance of feedback may be different in other experiments.

#### 5. Conclusions and future work

In this paper we studied the effect of relevance feedback processing on the success of IR methods for RT. We found out taking into account limited user feedback results in significant increases in both precision and recall. We modified feedback methods in both language model (MM and eMM) and vector-space model (SR and nSR) and mainly compared the performance of SR and eMM. The initial experiment results show that the feedback methods allow us to achieve a good precision and recall. eMM feedback method is generally more stable than SR, but SR can get better precision. By comparing performance of MM and eMM, nSR and SR respectively, we find that negative feedback information plays an important role in improving RT. This finding gives inspiration to analyst in practice. In the future, we will conduct more experiments to verify these findings. And study of the work of analysts in RT is also need to be done.

**Acknowledgements.** This work is supported by the National Natural Science Foundation of China under grant No.90718042, No.60803023; the National Basic Research Program (973 program) under grant No.2007CB310802; the National Hi-Tech Research and Development Plan of China under Grant No. 2007AA010303

#### 6. References

- [1] J.H. Hayes, A. Dekhtyar et al., "Helping Analysts Trace Requirements: An Objective Look", IEEE International Requirements Engineering Conference, Sept. 2004, pp. 249-261.
- [2] J.H. Hayes et al., "Improving Requirements Tracing via Information Retrieval", IEEE International Requirements Engineering Conference, Sept. 2003, pp.138-150.
- [3] J.H. Hayes et al., "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods", IEEE Transactions on Software Engineering, v.32 n.1, 2006, pp.4-19.
- [4] J. Cleland-Huang et al., "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability", IEEE International Conference on Requirements Engineering, Sept. 2005, pp.135-144.
- [5] G. Antoniol et al., "Recovering Traceability Links between Code and Documentation", IEEE Transactions on Software Engineering, Oct. 2002, pp. 970-983.
- [6] Fei Song, W. Bruce Croft, "A General Language Model for Information Retrieval", International ACM SIGIR conference on Research and development in information retrieval, Aug, 1999, pp. 279-280.
- [7] Chengxiang Zhai, John Lafferty, "Model based Feedback in the Language Modeling Approach to Information Retrieval", Int. Conf. on Information and knowledge management, October, Atlanta, Georgia, USA 2001.
- [8] A. Figueroa, "Smoothing Methods for LM in IR", [www.lsv.unis-aarland.de/Seminar/LMIR\\_WS0506/LM4IR\\_slides/Alejandro\\_Figuera\\_Smoothing\\_Methods\\_for\\_LM\\_in\\_IR.ppt](http://www.lsv.unis-aarland.de/Seminar/LMIR_WS0506/LM4IR_slides/Alejandro_Figuera_Smoothing_Methods_for_LM_in_IR.ppt).
- [9] S.K.M. Wong et al., "Generalized Vector Space Model In Information Retrieval", International ACM SIGIR conference on Research and Development in Information Retrieval, 1985, pp.18-25.
- [10] G. Salton and C. Buckley: Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management, 1988, pp. 513-523.
- [11] C.T.YU et al., "A Statistical Model for Relevance Feedback in Information Retrieval", Journal of the ACM (JACM), 1976, pp.273-286.
- [12] G. Salton, Chris Buckley, "Improving Retrieval Performance by Relevance Feedback", Journal of the American Society for Information Science, 1990, pp.288-297.
- [13] Andrian Marcus et al., "Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing", Int. Conf. on Software Engineering, 2003.
- [14] Gospodnetic O ,Hatcher E, Lucene in Action, Manning Publication, 2006.
- [15] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval. Addison-Wesley, 1999.
- [16] J. J. Rocchio: Relevance feedback in information retrieval. In The SMART Retrieval System:Experiments in Automatic Document Processing, 1971, pp.313-323.
- [17] Xuanhui Wang et al., "A study of methods for negative relevance feedback", SIGIR'08, July. 2008.
- [18] Telelogic DOORS, <http://www.telelogic.com>

# Applying Transformation Rules to Improve i\* Models<sup>‡</sup>

Márcia Lucena<sup>1,2</sup>, Carla Silva<sup>2</sup>, Emanuel Santos<sup>2</sup>, Fernanda Alencar<sup>3</sup>, Jaelson Castro<sup>2</sup>

<sup>1</sup>*Departamento de Informática, Universidade Federal do Rio Grande do Norte, Brasil*

<sup>2</sup>*Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil*

<sup>3</sup>*Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco, Brasil*

{mjnrl, ctlls, ebs, jbc}@cin.ufpe.br, fmra@ufpe.br

## Abstract

*Requirements engineering (RE) is considered a key activity in almost all software engineering process. i\* is a goal-oriented approach widely adopted in the RE, as it offers a modeling language that describes the system and its environment in terms of actors and dependencies among them. Often i\* models become cluttered both for small and large software systems, compromising their evolution and understandability. In this paper we propose to use model transformation rules and a systematic process to increase modularity and, therefore, comprehensibility and scalability of i\* models. To evaluate our approach, we use metrics to assess the complexity of i\* models before and after applying our approach in an e-commerce case study.*

## 1. Introduction

The i\* framework [1] is a goal-oriented requirements engineering approach widely used in academy and industry [6]. It captures the social and intentional relationships in the system organizational environment as well as some quality attributes and functionalities of system. This framework has a rich ontology that contains many constructors to create its models. However, as the complexity of the problem at hand grows, the i\* models may become cluttered, decreasing its understandability and scalability.

To reduce complexity of the models and improve comprehensibility of software artifacts, we can use decomposition mechanisms that divide software into meaningful and manageable pieces, by using the divide and conquer principle. Although i\* incorporates a decomposition mechanism based on strategic actors, it has not been properly explored to reduce the models complexity. This mechanism could be used to

decompose the actor representing the system into sub-actors that are easier to understand and manage.

In this work we propose a systematic way to handle the complexity of i\* models. We advocate the decomposition of actors by the use of model transformation rules. A transformation rule is a central concept of many model transformation approaches [2], since it usually describes the logic behind the transformation itself. The proposed transformation rules will produce i\* models semantically equivalent to the original models, but simpler and easier to understand. After applying them, the i\* actors become simpler and the i\* models become easier to grasp. The evaluation of i\* models complexity can be performed by the use of adapted McCabe's [3] and Halstead's [4] metrics.

This paper is organized as follows. Section 2 overviews the i\* framework and motivates our work by using an e-commerce example. Section 3 presents our approach to reduce i\* models complexity. Section 4 illustrates the use of our approach and presents a brief evaluation of this exercise. Section 5 summarizes our work and points out open issues.

## 2. Motivation

i\* models describe both the system and its environment in terms of a set of actors linked by dependencies among them. There are two different abstraction levels in i\*: the Strategic Dependency (SD) Model and Strategic Rationale (SR) Model.

In order to illustrate i\* models, let us consider the Medi@ system case study presented in [5]. Medi@ is a front-store on Internet to sell and ship different kinds of media items. Figure 1 shows a fragment of the SR model for the Medi@ actor, presenting the expanded view of this actor. The highlighted (a, b, c) parts present in Figure 1 will be discussed in sections 3 and 4.

---

<sup>‡</sup> This work was supported by CNPq and CAPES research grants and BIT initiative.

In SD model, an actor can depend upon another one to satisfy a goal, execute a task, provide a resource, or satisfy a softgoal. Softgoals are usually associated to non-functional requirements, while goals, tasks and resources are associated to system functionalities [6].

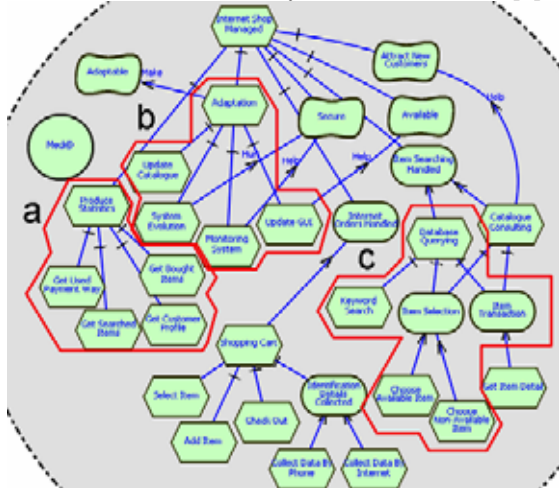


Figure 1. The Medi@ SR Model

The SR model is used to (i) describe the interests and motivations of the participants in the process, (ii) enable the assessment of possible alternatives in the definition of a process, and (iii) detail the existing reasons behind the dependencies among the actors. To support the analysis of opportunities and vulnerabilities for different actors, SR models include intentional elements such as goals, tasks, resources and softgoals, as well as three new types of relationships: means-end, task-decomposition and contribution link.

A task-decomposition link is a relationship between a task and its parts, which describe how to perform this task. In Figure 1, the Medi@ actor specifies a root task Manage Internet Shop that is firstly refined, through task-decomposition links, into intentional elements. These elements are further refined by using task-decomposition, means-end or contribution links, to discover the Medi@ system requirements.

A means-end link indicates a relationship between an “end” and a “means”, wherein a “means” is an alternative and usually an “end” is a goal to be achieved [1]. Considering the Media@ actor (Figure 1), there is a means-end link from Choose Available Item task (“means”) to Item Selection goal (“end”).

The contribution link describes a contribution of a “means” (task or softgoal) to the achievement of an “end” (softgoal). This link provides a qualitative reasoning using a multi-valued evaluation scheme to represent the contribution (e.g., Help, Hurt, Make) [7]. In Figure 1, the Update GUI task contributes positively (Help) to the satisfaction of the Available softgoal.

Using all these constructs to analyze, discover and specify the system requirements, contributes to produce  $i^*$  models loaded with information that captures characteristics of both the system organizational environment and the software system itself. Thus, the more detailed  $i^*$  models are, the more complex they become, mainly due to the refinement of the system actor. This complexity can be reduced by using decomposition mechanisms that divide complex actors into meaningful and manageable sub-actors.

### 3. An Approach to Reduce Complexity

To reduce the complexity of the  $i^*$  models, we propose a process composed of three principal activities: (i) Evaluation of  $i^*$  Models; (ii) Analysis of System Actor; (iii) Application of the Model Transformation Rules. To perform these activities, it is required to use, respectively: (a) metrics to assess the complexity degree of the initial  $i^*$  models, (b) conditions to guide the system actor’s decomposition, and (c) model transformation rules to generate simpler  $i^*$  models. At the end of this process the metrics of the item (a) are used to assess the complexity level of the resulting  $i^*$  models. This process is semi-automatic, since decisions are likely to be taken by the requirements engineer.

#### 3.1 Evaluation of $i^*$ Models

We start the process by evaluating  $i^*$  models using the McCabe’s metrics for cyclomatic complexity [3] and Halstead’s metrics for volume [4]. The cyclomatic complexity measures the number of independent paths in directed graphs, such as  $i^*$  models. Since this metric does not consider size as a parameter, we also adopted the volume metric. This metric measures the amount of information contained in a model in terms of the total number of links and elements, and the number of distinct links and elements. This assessment helps to the development team decide if the  $i^*$  model is complex and if it is necessary to follow to next activity. After the assessment, if the development team judges the  $i^*$  model as too complex, according to the resulting of the metrics and their experience in other projects, the second activity of the process can be performed.

#### 3.2 Analysis of System Actors

The decomposition criteria used in this approach is based on the separation and modularization of elements that are not strongly related to the application domain and, therefore, can be easily reused in different domains. For example, in the  $i^*$  model presented in Figure 1, that captures the Medi@ system requirements

and their relationships with the stakeholders, the requirement engineer can identify those elements that are uniquely related to the application domain (e-commerce) and those that are not. For example, the sub-graphs highlighted as regions ‘a’, ‘b’ and ‘c’ of Figure 1, are independent from the e-commerce application domain and, therefore, they can be moved from the main system actor (Medi@) to another (new) system actor. After doing this, the resulting model will present more system actors and these actors become dependent on each other. Besides, the contextual information present in this resulting model must be semantically equivalent to the original model.

At the end of this activity, the actors representing the system modularize fewer internal elements, becoming less complex and easier to understand and maintain. Furthermore, this approach also aims at promoting reuse of system actors. Since the new system actors are independent from the application domain, they may be present in the specification of system requirements of another domain. Thus, separating the independent elements in other actors can improve reusability and maintainability of system specification at the requirements level. In fact, the example presented in Figure 1, illustrates that the elements related to statistics production functionality (highlighted sub-graph ‘a’) could be used as part of a system from a different application domain. The same rationale could be applied to the other two sub-graphs highlighted in Figure 1 (sub-graph ‘b’ and ‘c’).

To assist the requirements engineer in decomposing the system actor, we have formulated the following (pre) conditions: (C1) Find internal elements in the system actor that are independent from the application domain; (C2) Check if these elements can be moved from the original model to another actor without interfering with the behavior and comprehensibility of actor’s internal details; (C3) Check if these elements could be reused in different application domains.

### 3.3 Application of the Transformation Rules

In this activity a suitable model transformation rule must be chosen based on the type of relationship between the elements to be moved and the elements that will remain in the original system actor. These transformation rules are applied to elements of type goals, softgoals and tasks, because they can be further refined. Since some elements are selected in previous activity, the elements that will be treated first are those have less impact on the actor. In this case, we are considering that the elements closer to the leafs have less impact when they are moved to another actor.

The proposed transformation rules aim at delegating internal elements from the system actor to other actors. This delegation must ensure that the new actors and the original actor establish a dependency relationship.

**Table 1: TR to move sub-elements**

TR1 - Move a sub-element in a task-decomposition
<p style="text-align: center;"><b>Original Model</b></p> <p style="text-align: center;"><b>Target Model</b></p>
<p><b>Pre-condition:</b> A root task of a graph is decomposed into sub-elements (tasks or goals) that are also root of sub-graphs, but do not share any sub-element through task-decomposition or means-end links.</p>
<p><b>Effects:</b> The sub-graph that is independent from application domain (e.g. the sub-graph whose root element is the Task 2) is moved from the original actor to a new actor. This new actor has the same name of the root of the transferred sub-graph. This root will be replicated as a dependency relationship of same type relating the original actor, as the depender, and the new actor, as the dependee. Besides, all the existent external dependencies with this transferred sub-graph will be transferred to the new actor.</p>

As shown in Table 1, the transformation rule is structured to show information such as (i) the name of the rule; (ii) a figure to illustrate the context in which the original model can match before applying the rule; (iii) a figure to illustrate the resulting model after applying the rule; (iv) a description of the pre-conditions for the rule to be applied, and (v) a description about the effects produced by the rule.

TR1 shows a transformation rule that moves a sub-element present in a task-decomposition to another actor (Table 1). This transformation rule was defined based on a property found in the i\* framework and related to the actor’s boundary. This property states that the semantics of an outgoing dependency link from a task is equivalent to the semantics of a task-decomposition link, that is, the outgoing dependency behaves as sub-component of that task inside the depender actor [1]. Also, in this transformation, all the intentional elements present in dependencies entering in the new actor are replicated inside that actor, as occurs in the extended version of i\* used in Tropos [8].

TR2 shows a situation (Table 2, original model) where the sub-graph to be moved has the root as a

“means” in a means-end relationship. In this case, this sub-graph is moved to a new actor, the root element is replicated both inside the original actor and as a dependency from the element inside the original actor to the root of the sub-graph moved to the new actor.

**Table 2: TR to move alternatives**

TR2 - Move “means” sub-graph in a means-end link
<p style="text-align: center;"><b>Original Model</b></p>
<p style="text-align: center;"><b>Target Model</b></p>
<p><b>Pre-condition:</b> A root goal of a graph is an “end” in one or more means-end relationships and at least one of its “means” is a sub-graph (alternative) that does not share any element (through task-decomposition or means-end links) with other sub-graphs (independent sub-graph).</p>
<p><b>Effects:</b> Each independent sub-graph (alternative) will be moved to a new actor with the same name of the sub-graph’s root. The roots of the transferred sub-graphs must be replicated inside the original actor to keep the original mean-end relationship. From each of these replicated elements, a new dependency of the same type and name must be created from the original actor to the root of the sub-graph moved to the new actor.</p>

After applying rules TR1 and TR2, it can occur that the resulting model is not in conformity with the *i\** notation suggested by the Istar Guide [7]. For example, a contribution link in the original model can result in a crossing relationship from an actor to another (see Table 3, original model). In this case, we need to use a corrective rule, such as TR3. TR3 suggests replicating the softgoal involved in the contribution relationship both inside the original actor, and as a softgoal dependency outgoing from the new actor to the original actor. This rule was defined to preserve the information about contribution links and maintain the information about contribution links and coherence of *i\** models as it is proposed in [9].

TR4 (see Table 4) is applied when the sub-graph to be moved out has a sub-element shared (Task 3 in the Original Model presented in Table 4) with other sub-graphs. One of the previous rules, TR1 or TR2, is

applied to move a sub-graph (in this case, the sub-graph with Goal 1 as root) to another actor.

**Table 3: TR to move a contribution link**

TR3 - Contribution link crossing actor’s boundary
<p style="text-align: center;"><b>Original Model</b></p>
<p style="text-align: center;"><b>Target Model</b></p>
<p><b>Pre-condition:</b> There are elements such as task, goal or softgoal contributing to the achievement of softgoals that are out of the actor’s boundary.</p>
<p><b>Effects:</b> A softgoal element, with the same name, must be created inside the actor from where the contribution link outgoing, to keep the contribution link inside that actor. A softgoal dependency, with the same name, must be created from the new softgoal to the other softgoal, with the same name, inside of the other actor.</p>

At this point, TR4 suggests a priority policy to choose with which sub-graph the shared element (Task 3) will stay: (i) check the types of the roots in the sub-graphs sharing the element. The sub-graph whose root type has the higher priority will keep the shared element. The priority for root element type is goal, softgoal and task, in this order; (ii) if all sub-graphs’ roots are of the same type, check the position of the roots in the sub-graphs sharing the element. The sub-graph’s root that is closer to the root of the overall graph will keep the shared element; (iii) if all sub-graphs’ roots have the same type and are in the same level in relation to the overall graph’s root, then the shared element stays with the sub-graph(s) that will remain in the original actor.

## 4. Running Example

After the requirements engineer has decided to apply the approach to reduce the complexity of the Medi@SR model (Figure 1), he can identify some elements that could be moved to other actors. For instance, considering the conditions C1 and C2 presented in the section 3.2, the Produce Statistics, Adaptation, and Database Querying tasks were identified. The Produce Statistics and Adaptation tasks represent issues that are not strictly related to the e-commerce application domain. Therefore, it can be moved from the original actor without interfering in its understanding and

original purpose. Similarly, the Database Querying task related to Item Searching Handled goal, by a means-end link, can be moved from Medi@ because it can be reused in other domains.

**Table 4. TR to move shared elements**

TR4 - Move a shared sub-element
<p style="text-align: center;"><b>Original Model</b></p>
<p style="text-align: center;"><b>Target Model</b></p>
<p><b>Pre-condition:</b> There is an element (Task 3) that is shared by different sub-graphs (through task-decomposition, contribution or means-end link) and at least one of these sub-graphs is moved to a new actor (rules TR1 or TR2).</p>
<p><b>Effects:</b> The shared element will remain in the sub-graph whose root element has the highest priority. The relationships with the remaining elements will be replaced by dependencies, as stated by the rules TR1, TR2 and TR3.</p>

To select the suitable transformation rules to these identified elements, we need to observe the type of relationship that these elements have with the remaining elements. For instance, TR1 should be applied to Produce Statistics task that is a sub-element of Internet Shopping Managed task, in a task-decomposition relationship. Afterwards, TR3 is applied to replace contribution links crossing actors' boundaries by softgoal dependencies, aiming at maintaining the coherence with  $i^*$  notation.

For the sub-graph in which Database Querying task is the root, TR2 is applied to move sub-elements that are alternatives (Database Querying task) to achieve goals (Item Searching Handled). Afterwards, TR4 is applied to move shared elements (Item Selection and Item Transaction goals are also sub-elements of Catalogue Consulting task). Figure 2 shows the resulting model after applying the proposed process.

After modularizing the Medi@ actor (Figure 2), we evaluate the original and the resulting models in relation to the metrics presented in section 3.1. The Cyclomatic Complexity metric has shown that the complexity of  $i^*$  models was reduced in approximately 57%. This result indicates that the approach promoted a reduction of complexity in  $i^*$  models for the Medi@

system. Indeed, part of the main graph inside the system actor was modularized by other actors, reducing the number of graph ramifications inside only one actor. This strategy increases the total number of elements and links in the model, since 3 new actors and 5 new dependencies have been added. As result, after applying the Volume metric, it was observed that the global volume of the resulting  $i^*$  model increased from 362.11 to 523.05 (44.45%). This global increment of volume does not compromise the benefits of our approach, since the new model consists of a set of new actors that now divides part of the complexity initially concentrated in a single system actor. In fact, the volume of Medi@ actor decreased from 362.11 to 175.69 (51.48%), meaning that the remaining volume was transferred to the other actors, namely Adaptation, Database Query and Produce Statistics. Their respective volumes are 95.18, 69.19 and 36.00. In comparison to the initial volume of Medi@ actor, the volumes of the four actors present in the new model are smaller than 50% of that initial volume. This means that the model complexity was reduced by delegating responsibilities of a complex actor to other actors, thus helping a requirements engineer to better manage and maintain  $i^*$  models, since they can focus on different and simpler parts of the problem each time.

## 5. Related Work

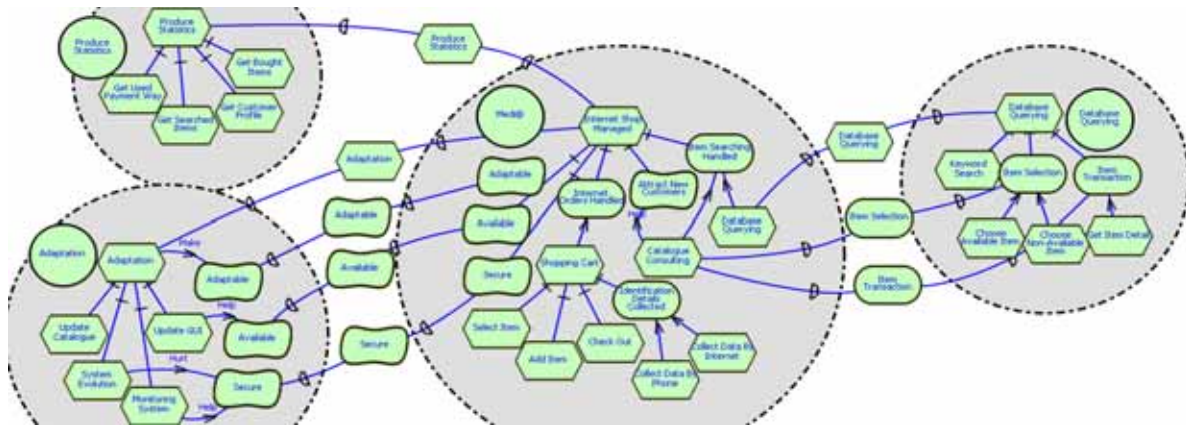
In [10] a systematic method to deal with scalability issues of  $i^*$  models is presented. The method reformulates the  $i^*$  framework to provide the concept of view - a projection over a model according to some criteria. Views were used as a way to divide one baseline model into self-contained segments in order to increase understandability of  $i^*$  models. Each view is associated with a formally defined selection rule to allow automating the projection of a specific view.

In [11] the authors use the principles of Aspect-Oriented Software Development [12] to simplify  $i^*$  models. Their approach identifies, modularizes and composes crosscutting concerns in  $i^*$  models. They extended the  $i^*$  modeling language by adding aspect-oriented abstractions. Their aim was reducing the graphical complexity of  $i^*$  models.

In [13] was conducted an exploratory study to identify the impact of applying a catalogue of patterns to modify  $i^*$  models. New concepts, that are part of a catalogue of patterns, were added to modify the model.

Although they had gains in other model attributes, no reduction of models complexity was observed. Our approach also proposes using iterative modifications in  $i^*$  models.





**Figure 2. Resulting Strategic Rationale Model**

However, it does not require the addition of new concepts because it uses mechanisms of delegation and transformation rules to divide the system actor in new actors. To select the elements to be delegated, we use a semi-automatic process.

Tropos [5] relies on *i\** models in several stages of software lifecycle and uses the relationship *is-part-of* to decompose system actors. But, different from our approach, no systematic way was proposed for its use.

## 6. Conclusions

A process to handle the complexity of *i\** models was presented in this paper. This process proposes to balance the responsibilities of a system actor, delegating them to other (new) system actors. A semi-automatic process can guide the evaluation of *i\** models' complexity, the use of (pre) conditions to choose which part of the system actor can be delegated to another (system) actor and the selection among a set of transformation rules to modify *i\** models. These rules create new actors, move parts of a system actor to these new actors, and ensure that the resulting model is semantically equivalent to the original model.

Currently we are carrying out experiments with metrics to assess separation of concerns. Besides, qualitative experiments will be used to evaluate what is the optimal size of SD models. With these values, we can determine when an actor is considered complex according its size. Other future work includes formalizing these transformation rules in a transformation language based on Eclipse framework and designing architecture from simpler *i\** models.

## 7. References

[1] Yu, E., "Modeling Strategic Relationships for Process Reengineering", Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.

[2] K. Czarnecki, S. Helsen, "Classification of Model Transformation Approaches", 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, Anaheim, CA, USA, 2003.

[3] T. McCabe, "A Complexity Measure", IEEE Trans. of Software Engineering, 2(4), 1976, pp. 308-320.

[4] Halstead, M. H. "Elements of Software Science (Operating and programming systems series)", Elsevier Science Inc., New York, NY, USA, 1977.

[5] J. Castro, et al., "Towards Requirements-Driven Information Systems Engineering: The Tropos Project", Information Systems Journal, 27(6), 2002, pp. 365-389.

[6] E. Yu, J. Castro, A. Perini, "Strategic Actors Modeling with *i\**", Tutorial Notes, 16th Intl. Conf. on Requirements Engineering, IEEE Computer Society, Spain, 2008, pp.01-60.

[7] Grau, G. et al., 2008. *i\** Wiki Home, In <http://istar.rwth-aachen.de/tiki-index.php?page-ref-id=53>, 02/04/09.

[8] P. Bresciani et al., "Tropos: An Agent-Oriented Software Development Methodology", Journal of Autonomous Agents and Multi-Agent Systems, 8(3), 2004, pp. 203-236.

[9] Horkoff, J., Using *i\** modeling for evaluation, Master thesis, Department of Computer Science, University of Toronto, Canada, 2007.

[10] You, Z., "Using Meta-Model-Driven Views to Address Scalability in *i\** Models", Master thesis, Department of Computer Science, University of Toronto, Canada, 2004.

[11] F. Alencar, et al., "Integration of Aspects with *i\** Models", Agent-Oriented Information Systems IV, LNCS, Vol. 4898, Springer-Verlag, 2008, pp. 183-201.

[12] A. Rashid, et al., "Modularisation and Composition of Aspectual Requirements", 2nd Intl. Conf. Aspect-Oriented Software Development, ACM Press, 2003, pp. 11-20.

[13] M. Strohmaier et al., "Can Patterns Improve *i\** Modeling? Two Exploratory Studies", REFSQ'08, LNCS, Vol. 5025, Springer-Verlag, France, 2008, pp. 153-167.

# NESTED NL REPRESENTATION FOR OO ANALYSIS AND DESIGN

Magda G. Ilieva, Olga Ormandjieva  
Department of Computer Science and Software Engineering  
Concordia University, Montreal, Canada  
magda\_i@yahoo.com, ormandj@cse.concordia.ca

## ABSTRACT

This article proposes a methodology for building software engineering (SE) models from unrestricted natural language (NL). The method is based on a nested representation of NL, which is a superior structural technique to other known object-oriented (OO) techniques, like the noun-object, verb-method, and modifier-property analogies. The novelty of our approach is that it shows the importance of the structure of concepts represented through language, and not through their grammar or organization.

## KEY WORDS

Knowledge representation, Nested representation of text, Concept structure approach to NLP, NLP approach to SE

## 1. INTRODUCTION

As the complexity of software systems has grown, so have the challenges facing software analysis, design, and programming. These new demands have given rise to new applications, theories, and technologies. Many of them focus on the analysis of NL description of requirement specifications, mainly because there have been shown to be analogies between OO concepts and NL. For this reason, the focus of computer linguists has turned to SE, and their work has led to satisfying results: the continuous creation of theories and tools which translate, partially or completely in a semi- or fully automatic way, the NL description of a software system under consideration into a formal or semiformal representation. Moreover, the work of these linguists has been stimulated and facilitated by the use case-driven development and the Unified Modeling Language (UML), some features of which place it between informal NL description and its formal SE model:

- Some of the questions posed during the process of building a Use Case model, like *How will the application be used?* or, more precisely, *What are its functional requirements?* are similar to linguistic (NLP) questions, like *Which are the actors?* and *What do they do?* Various developments based on NLP technology consider this UML model [6].
- A linguistic analogy can also be found for an activity diagram or a sequence message chart. The questions: *What is the activity? To where is the activity directed? What is the object of the action and how is it changed thereafter? What is the consequence of the activities over time? What are the conditions leading to their execution?* are posed in an NLP approach to model creation, as described in [1,2,3].
- The most important question for an OO class diagram is: *How are classes found?* Linguistic analysis is a very

popular method among those available to answer this question, and is, in fact, represented in all guidelines for finding classes [4,5]. We offer here one possible answer to the question of how to obtain an OO class model from NL requirements automatically.

The paper is organized as follows: Section 2 outlines the contributions of this research. The methodology is explained in section 3, and an illustration of a case study is presented in section 4. The conclusions and future work directions are outlined in section 5.

## 2. OUR CONTRIBUTION

Our work and research in the field of the NLP approach to SE modeling cover the following topics related to the next generation of graphical models: the OO class Diagram [7], Use Case Paths [10], the Hybrid Activity Diagram [8], and the Domain Model [11]. We have developed and published methodologies for the creation of these UML and similar models, which are based on three basic formalisms for NL representation – tabular [7], graphical [9], and nested [11]. In the three cases, we use the relation as a main notion for the analysis and a building element in the creation of the SE model. We structure the relations in NL on different levels. In NL, and also in the tabular and graphic representation of NL, the main relation is the predicate (verb phrase) between the subject and the object (noun phrases). No matter how complex the sentence, i.e. how many predicates it contains, each predicate is placed in a separate row in the table, and the Su(bject) and Ob(ject) linked to it are placed on its left-hand and right-hand side respectively (see Table2). Some Su or Ob cells can remain empty. For example, in many cases, the passive verb and imperative sentences (see case study) have no Su. However, this does not change the fact that a predicate relation exists anyway. The Su and Ob are identified by understanding the context, which is why they have not been explicitly indicated. On their own, Su and Ob can also be represented as relations between concepts, which we call structures of concepts or structural relations: prepositional, noun-noun modifier, adjective-noun modifier, enumerative. As with the relations in a simple sentence, high-level relations exist between predicative relations (two or more simple sentences combined into one complex sentence). The type of complex relation is defined by the type of conjunction between them. Examples of complex predicate relations are: IF-THEN sentences, relative sentences, and simple sentences connected by a conjunction. Fig. 1 shows a generalized scheme of NL relational structures.

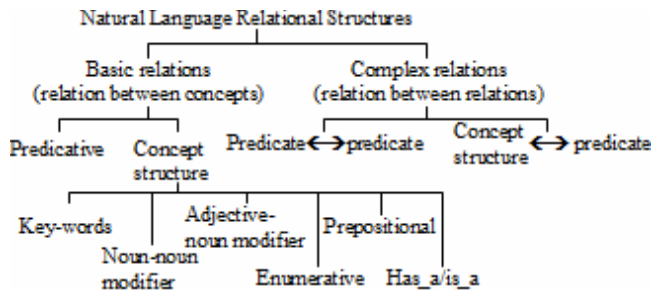


Fig. 1. Generalized scheme of NL relational structures

### 3. METHODOLOGY

**For the Domain Model (Ontology):** Working on the structural relations, as well as the operations on the structures, we arrived at the nester representation (NR) of concepts, which is suitable for the creation of a domain model (DM) or ontology [11]. Topics for consideration in this type of representation are only those structural relations in NL that reflect possession. To the group of noun-noun and adjective-noun modifiers (*has-a* and *is-a* relations), we can add prepositional phrases with prepositions for location and possession: *of*, *in*, *into*, etc. We will explain the nested representation shortly using examples, but for now: the term *NL requirements document* means that the document consists of requirements expressed in NL. The nested representation will be styled *document(requirements(NL))*. The phrase *ambiguity in NL and TLG* means that NL and TLG (Two-Level Grammar) has an ambiguity and will be represented as *(NL,TLG)ambiguity*, which, using the mathematical operation of removing brackets, can be represented as *NL(ambiguity),TLG(ambiguity)*. Brackets and the comma are the only operations that we use in a nested representation. Grouping operations can be applied on nested structures, and, when all the structures are grouped together, we obtain a hierarchical structure of the DM/Ontology of the problem domain, extracted from the text description. The grouping operations are based on the matching principle. Two nested structures are matched when their heads are equal. Then, a new structure will emerge from a merging (grouping) of the two structures which will have a common head and body containing the merged bodies of the grouped structures. The pseudocode of the algorithm for merging nested structures can be found below and an online demonstration of it can be tested at the following address: [www.nlping.com/nestedStr/prophp.php](http://www.nlping.com/nestedStr/prophp.php).

```
function merge ( Hi(Bi),Hj(Bj) ) {
if ( H=match(Hi,Hj))
    newStr=H(merge(Bi,Bj))
else { if (H=match(Hi,Bj))
        newStr=Hj(merge(Hi(Bi),Bj) )
        else (H=match(Hj,Bi))
            newStr=Hi(merge(Hj(Bj),Bi) );
return newStr;}
function match( Hi(Bi),Hj(Bj) ) {
if ( Hi=Hj) return true ; else return false;}
```

An example explaining the above algorithm follows.

Vehicle (authorized, nonAuthorized)	(1)
Vehicle (authorized (driver) )	(2)

**Result:** Vehicle(authorized(driver), nonAuthorized)

Structures (1) and (2) have the same head, *Vehicle*, and the bodies B1 = *authorized, nonAuthorized* and B2 = *authorized (driver)*. In the step1 of the recursion, the common head *Vehicle* is defined and B1, B2 are passed for new matching. This time, the match function finds 2 structures (one from each body) with the same head: *authorized* and *authorized(driver)*. The two bodies, [empty] and *driver*, merge and are added to the common head, which is *authorized*. The result, *authorized(driver)*, is added to the remaining part of the body (*nonAuthorized* in this case), which is attached to the head defined in step1.

**Extended Nested Representation:** While brackets and the comma are operators that are sufficient to represent the DM/Ontology models, where the relation ‘contain’ is basic, these operators are not sufficient for the solution of other problems, such as those from the mathematical domain. We consider the language of mathematics as a subset of NL with the following specificity: mathematical problems contain concepts which are arranged in relations like: before, after, between, from-to, less than, smaller than, equal to, etc. In NL, there are words which have procedural mathematical equivalents, like *difference* and *sum*. For these specific relations and procedural words, we will use their mathematical indications. The basic nested representation of text broadened with operators typical for the NL of a given problem domain is called Extended Nested Representation (ENR). We now show the applicability of the ENR of text for OO analysis and design on an example from mathematical problem domain [12].

### 4. CASE STUDY

The text is taken from “an informal but precise English description” and presents a problem defined in the following manner: *Write a function subprogram that, given two dates in the same year, returns the number of days between the two dates.*

The definition of the problem serves only to inform the reader and has not been analyzed, either in the source or in our solution. The aim of the author of the source is to obtain an ADA program from an NL text description, through a guided analysis by a human being. Our aim is to obtain an automated semiformal OO programming code through an algorithm and the ENR. The text is the following:

1. *If two given dates are in the same month, the number of days between them is the difference between their days of the month.*
2. *If the two given dates are in the different months, the following is done:*
  - (a) *Determine the number of days from the earlier date to the end of its month. Keep track of that number in a counter called the “Day\_counter”.*
  - (b) *For each month, starting from the first month after the earlier date and ending with the last month before the later date, add the number of days in that month to the Day\_counter.*

(c) Add the day of the month of the later date to the Day\_counter. Return that final sum as the number of days between the two dates.

The solution consists of 4 parts. The first two parts involve NL analysis and design, and the next two SE analysis and design.

#### 4.1. CASE STUDY NL ANALYSIS

This part of the solution goes through the following processing steps:

##### 1) The first step is to define the operators for ENR

According to our understanding of the character of NL, it consists of structures. Prepositions are a very important part of the creation of these structures. They connect the concepts to the structures, and that is why we consider them as operators. In the following table, we have generalized the preposition-operators that are found in this case study.

Table1

Operators	NL equivalent	Example	ENR
possession	of,in,into,has a,is a	A in B	B(A)
enumeration	keyword/orthography	two days	d1,d2
bounded content	from-to, between, starting-ending	from1 to2	1--2
precede-follow	before-after	2 after 1	2>1
condition-consequence	if-then	If A then B	A=>B
assignment operation	add to, subtract from	add A to B	B+=A
equality	equal,named,called,as	A is called x	x=A

The interpretation of the operators *from-to*, *starting-ending*, and *between* is similar, in that they define the content between two positions. While *from-to* defines the starting and ending points, *between* uses a plural noun as an argument, which means that there can be more than two points. The mathematical equivalent of these two operators, 'between(A,B)' and 'from A to B', is: i)  $\geq A$  and  $\leq B$ , if we consider one value between two boundary values; or ii)  $A--B$ , if we consider more than one value. The operators *after* and *before* also have a similar meaning. *After* A defines a value following position A and has the mathematical equivalent  $>A$ , while *before* A defines a value before position A and has the mathematical equivalent  $<A$ .

##### 2) The next process is structuring and normalizing the text

First, we represent the text in tabular form. Then, we normalize it through processing, which makes it suitable for formal representation; for example, finding the references, removing repetitions, checking for inconsistencies, etc.

##### 3) Finally, on the prepared in that way structured text, we apply ENR

Since the tabular representation, normalization, and the nested representation are linked, we will address them consecutively for the different parts of the text, which contains 5 paragraphs.

**Point 1:** The following table displays the structured representation of the text from the first paragraph.

Table2

Cn	Subject	Pr	Object	Con
If	the two given dates	are	in the same month	,
	the number of days between them	is	the difference between their days of the month	.

In order to address the reference issue, we use the principles of correspondence and proximity, and the template for singular/plural consistency. For example, we define *them* and *their* as referring to *dates*.

The first sentence contains an IF-THEN operator, with the following structures in each part:

(1.1) The IF part contains the operator *are in*, which links two structures: enumerative, *two given dates*, and adjective-noun modifier, *the same month*. The sense of the operator *is in* is the same as that of *has* and the interpretation of the phrase is: (*the same month*) HAS (*two given dates*). The nested representation will be:

$month\_same(dates1,dates2)$  (1.1)

(1.2) The THEN part contains the operator *is*, which links two structures:

- The first structure, (*the number of days*) between *dates*, actually represents two structures, one containing the operator *of*, and the other containing the operator *between*. The operator *of* means that the concept on its right-hand side contains the one on the left-hand side, and *the number of days* means that *the days* has a number and will be represented as  $days(number)$ . The operator *between(dates)* can be represented as  $date1--date2$ . The nested representation of whole group is:  $(date1--date2)days(number)$  (1.2a)

- The second structure, *the difference between (dates days of the month)*, consists of the simple concept *the difference* and the operator *between*, which has the complex structure *of* as an argument: ( $dates\ days$ ) *of* (*the month*), the representation of which will be  $month(dates(days))$ . Applying the operator *between* on the structure defined in this way, we obtain:

$(month(date1(days)--month(date2(days))))difference$  (1.2b)  
Combining the different parts of the IF-THEN operator, we obtain:

**IF**  $month\_same(date1,date2)$   
**THEN**  $(dates1--dates2)(days(number))=$   
 $(month(date1(days)--month(date2(days))))(difference)$

**Point 2:** This paragraph is also an IF-THEN operator, with the following parts:

The IF part is: (*the two given dates*) *are in* (*the different months*). As with the arguments regarding the sense of *are in* from point1, this phrase can be represented as:

$IF\ month\_different(date1,date2)$  (2.1)

The THEN part: *the following is done* is a verb phrase *done*, after which there is an enumerative structure prompted by the key word *following* and the orthography semicolon and three numbered points. From this phrase, we only keep the verb, and, for the entire IF-THEN sentence, we obtain:

**IF**  $month\_different(date1, date2)$  **THEN** do

**Case (a):** The following table shows the structure and the nested representation.

#	Pr	Ob	NR of Ob
1	Determine	the number of days	days(number)
		from the earlier date	date_earlier
		to the end of its month	date_earlier(month(end))

2	Keep	track of that number in a counter	counter(days(number(track)))
	call	"Day counter"	

When we combine the parts from the 'NR of Ob' column, we obtain: (2.1 a)

```
Determine{(date_earlier(month(end))--date_earlier)days(number);}
Keep { counter (day (number (track))= "Day_counter" } ;
```

Braces separate the verb/action from the concept structures. **Case (b)** is presented in the next table. The nested representation can be seen in the column 'NR of Ob'.

#	Pr	Ob	con	NR of Ob
1		For each month		Foreach(month
		starting from the first month		month first
		after the earlier date	and	>date earlier
		ending with the last month		month last
		before the later date		<date later
	add	the number of days in that month		month(days (number))
		to Day_counter		"Day_counter"

We replace the ENR operators (starting-ending, add-to) according to notation from Table 1, and, combining the different parts from the 'NR of Ob' column, we obtain:

```
Foreach((month_first >date_earlier)--
(month_last <date_later)month)
Day_counter += month(days(number));
```

**Case (c)** is structured in the next table.

#	Pr	Ob	NR of Ob
1	Add	the day of the month of the later date	date_later(month(day))
		to day counter	Day counter
2	Return	that final sum	sum final
		as the numbers of days	days(number)
		between the two dates	date2--date1

After replacement of the operators (add-to, as) and connecting the different parts from the 'NR of Ob' column, we obtain:

```
Day_counter += date_later(month (day)) ;
return { sum_final= ( date2--date1) days(number) } ;
```

What we need to do now is collect all the formulas (depicted in the solid table) and obtain the result of the analytical phase:

1	IF	month same (date1,date2)
2	THEN	(date1--date2)(days(number))= (month(date1(days))--month(date2(days))) difference
3	IF	month different (date1, date2)
4	THEN	do
5	(a)	Determine {(date_earlier (month (end)) --date_earlier) days (number) } ;
6	Keep	{counter (day (number (track))) = "Day_counter" } ;
7	(b)	Foreach ( ( (month_first>date_earlier) -- (month_last<date_later))month)
8		Day_counter += month(days (number)) ;
9	(c)	Day_counter += date_later(month (day)) ;
10		return { sum_final= (date1--date2) days(number) } ;

Listing 1

## 4.2. CASE STUDY NL DESIGN

Here, we are looking for analogies and similarities between the NL nested representation and the programming language.

On line 1 of Listing 1, we have defined the function *month\_same* with two arguments, *date1* and *date2*. The goal of this function is to verify whether or not the two dates fit in a single *month* and to return the result from that verification.

On line 2, an equality is defined. If we replace the left bracket with a dot, as in the OO style of programming, which means that method or property of the object, and we remove the corresponding right bracket, we will obtain:

$$(date1--date2).days.number= (month.date1.days--month.date2.days).difference \quad (2.2)$$

The judgment on which we are basing this change is the following: in a complex concept structure, the concepts are separated and structured through brackets, which means that the outer concept, from the left-hand side (parent), contains the inner concept, i.e. the one on its right-hand side (child). For example, *month(date2(days))* means that *month* has *date2*, which has *days*. The possession in OO-programming is written with a dot. We will also use it, and obtain the following: *month.date2.days*, which means that the object *month* with the method *date2*, and *date2* with the method/property *days*. With the notation defined in this way, let us consider equality (2.2). We have two expressions which are linked with the equals sign. We assume that this is not an algebraic equality, and most probably means that the left-hand side is a statement and the right-hand side is a formal expression, or implementation of that statement. This is why we can introduce a variable (*Result1*) for the definition of the left-hand side, or keep the corresponding NL expression unchanged, for example: *numberOfDaysBetweenDates*.

There is an additional point concerning the interpretation in (2.2). The method *difference* is applied at the same time as the operator *between* over the same expression. Since they have a similar semantic meaning, one of them is redundant. We have two options: i) remove the method *difference* and obtain *Result1*= (month.date1.days -- month.date2.days); or ii) remove the operation *between* (--), so that the operands become parameters of the method *difference*, and obtain *Result1*=difference(month.date1.days, month.date2.days).

On line3, the function *month\_different* is defined with two arguments, *date1* and *date2*. This function must check whether or not the two dates fall into different *months*, and return the result. The functions on line 1 and line 3, *month\_same* and *month\_different* respectively, are antonyms, and so have contrary meanings. One function can replace both, returning a result which is true in one of the cases, for example *month\_same*, and false in case of *month\_different*. We obtain the following code after the replacement:

1	IF	month same (date1,date2)
2	THEN	Result1=(month.date1.days--month.date2.days) ;
3-4	ELSE DO	

Line5: Determine {(date\_earlier(month(end)) -- date\_earlier days(number)}. After replacing the NL operators with software operators, we obtain:

Determine{(date\_earlier.month.end--date\_earlier).days.number}. We apply the method *days.number* to the two parts of the operator "--", which is equivalent to removing the brackets. We do this because it is logical, and because from mathematics we know that a method applied over the result of one operation is the same as applying the method over the operands, and after that performing the operation. We obtain the following expression: Determine{

date\_earlier.month.end.days.number--date\_earlier.days.number}

Line6:Keep {counter(day(number(track)))='Day\_counter'}. Over this code, we replace the left bracket with a dot and remove the corresponding right bracket. Then, on the left-hand side of the equation we have an expression, and on the right-hand side a named concept. The rules in programming and mathematics usually place these two entities in reverse order, the left-hand side containing the concept (in the role of a variable) and the right-hand side containing the expression that gives the variable a value.

Keep {"Day\_counter" = counter . day . number . track }

Lines 7-8: We perform the same processing as we did on line 6, replacing the left bracket with a dot, removing the corresponding right bracket, and removing the bracket around the operation "--". The sign "+=" on line 8 replaces the phrase 'add to', according to the ENR definition in Table 1. Our result is:

```
Foreach (month_first . month > date_earlier . month ) --
        month_last . month < date_later . month )
Day_counter += month . days . number ;
```

Line 9: Similar processing to that on line 8, which gives the result: Day\_counter += date\_later . month . day ;

Line 10: return{ sum\_final=(date1--date2) days(number)}. We remove the brackets, which means that we apply the method *days.number* over to *dates*. We obtain:

```
return{sum_final=date1.days.number--date2.days.number}.
```

Let us now put all the rows together to see the end result:

1	IF month_same (date1,date2)
2	THEN Result1 = (month .date1.days--month . date2 .days)
3-4	ELSE DO
5	(a) Determine { date_earlier . month . end . days . number--date_earlier . days . number};
6	Keep {"Day_counter" = counter . day . number . track };
7	(b) Foreach (month_first . month > date_earlier . month ) -- month_last . month < date_later . month )
8	Day_counter += month .days . number ;
9	(c) Day_counter += date_later . month . day ;
10	return{ sum_final = date1.days.number--date2.days.number};

Listing 2

### 4.3. CASE STUDY SE ANALYSIS

In the SE analysis phase, we have to define the structure of a software program, and, since we are using the OO approach, we have to define what the objects are, as well as their methods and properties. A nested representation of the requirements gives us hierarchies of the main concepts. This hierarchy is obtained automatically in a natural way only through examining the NL structures, and it corresponds to the hierarchy taught according to the

principles of the OO approach. The following basic consequences can be seen in our example:

month → date → days; month → days → number

date → month → days; date → days → number

A concept consequence is important because it defines the object and its methods and properties. From the defined order of the concepts, month→date and date→month, arises the question of whether *date* is an object having the property *month* or vice-versa. We have strictly adhered to the algorithm here, and any 'contradictions' come from the NL itself. Here are two phrases:

i)...the dates are in various months... → means that a *month* consists of *dates*, i.e. *month(date)*

ii)...from the earlier date to the end of its month... → *its* refers to *date*, i.e. *its month* ≡ *date's month*, i.e. *date* is an owner of *month*, i.e. *date(month)*.

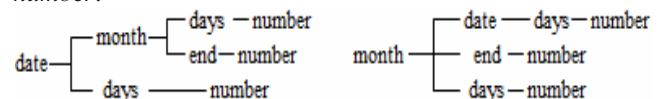
From the NL description, we can deduce that there are two objects: *month* and *date*. It seems that these objects also exist in the computer solutions. For example, *month(1.1.2009)* should turn into *January*. By contrast, *date(January)* should become *1 jan, 2 jan, ..., 31 jan*. They all satisfy the condition of being dates in January, i.e. these are all the dates in the month referred to. In other words, this is the number of days in the month.

This specific case, in which the consequence of the NL nested concepts gives rise to doubts as to the object-methods relation, does not compromise the applicability of the proposed algorithm, and different automated solutions can be proposed to resolve the contradiction. Among them are: i) previous experience, patterns; ii) a search of the Internet for code examples; and iii) the creation of statistical models for objects with a contradictory structure. A simple Google inquiry with the key words 'date object' and 'month object' returns 8 times more results than 'date object' alone.

### 4.4. CASE STUDY SE DESIGN

At this stage, we have to define what programming must do. To achieve this, we use the results from the previous phases (listing 2).

1) Object date with the methods *month*, *days*. On the next level, *month* is divided with two methods – *days* and *end*. All the leaf nodes of the *date* structure are of the type *number*.



2) There are five functions to be implemented from the listing:

```
function monthSame(date1,date2)
{ if (date1.month == date2.month) return true;
  else return false; }
function dateEarlier(date1,date2)
{ if (date1 < date2) return date1;
  else return date2; }
function dateLater(date1,date2)
{ if (date1 > date2) return date1;
  else return date2; }
function month_first (date)
{ return date.month; }
```

```
function month_last (date)
{ return date.month; }
```

3) The **foreach** construct is defined on line 7 of Listing2. Like the loop controls from the programming languages, it defines: iterator, initial condition, and final condition. In our case, *month\_first.month* plays the role of iterator. From an OO perspective, the expression is read: method *month* of the object *month\_first*. We presume that this is a tautology and leaves only *month*. The initial and final conditions are *date\_earlier.month* and *date\_later.month* respectively. With arguments defined in this way, we can rewrite lines 7-8 directly into the **for** control structure, which is widely used in every programming language:

```
for (month= date_earlier.month+1; month< date_later.month;
month++) { Day_counter = month .days. number }
```

4) The procedures/functions *determine*, *keep*, *add*, *return* have precise programming language equivalents, which is why we leave them unchanged.

## 5. CONCLUSIONS AND FUTURE WORK

**Evaluation:** In order to evaluate the effectiveness of our approach, we compare our solutions to a single example, obtained through various technologies. This is the result of comparing the solution in the cited source [12] and the solution presented here:

Russell Abbott solution	Our solution
8 transformation phases/12 pp	4 phases/6 pages
Human-guided process	Automated
NLP and SE processes are carried out in parallel; they influence each other.	Separates process results. NLP phase is independent of implementation.
Contains words/concepts that are not present in the text.	Contains only words/concepts from the text.
Keeps the NL operators for parent-child relations	NL operators are replaced with OO dot notation.

Below is a short description of another example from [13] solved with our approach.

Tagging with the syntax categories of the words helps us structure the text into a table:

1	Write	a program	
	to generate	1000 random numbers between 0 and 99 inclusive	.
2	You should count	how many of times each number	
	is generated		and
	write out	these counts to the screen	.

Applying the ENR to the text, we obtain:

```
write {program
{generate {Numbers(random(1000 (0- -99(inclusive)))) } };
shoud { count {eachNumber(times(howMany(generated))),
writeOut {counts=screen} } ;
```

What remains is to apply SE analysis and design skills, as well as write down the functions that have to be implemented. In the source, the solution looks like this:

```
for($i = 0; $i < 10000; $i++) { //comment }
foreach $count (@counts) { //comment }
```

*//comment* repeats the content of the text. The main effort in the approach cited in [13] is to reveal the relation between language key phrases and programming key words. Our

aim is to find a universal representation of NL, which in this project is a nested and extended nested representation based on the structural relations between the concepts. We consider NL to be built from structures, and in this way we link the syntax and the semantics.

**Summary:** We used the 'toy example' to demonstrate how the nested/operator representation of NL can serve for the translation of an NL description into a semiformal OO representation. Many authors correctly observe the OO character of the language, but, instead of tracking this object orientation, they take it outside language and try to explain it with grammatical terminology, such as common noun, proper noun, verb, attribute, etc. The structuring that is typical for the OO approach comes from the structured nature of things in the real world, which is reflected in NL. To build the SE model from unrestricted NL, we use the structure of the concepts in the text, rather than the grammatical structure of phrases.

Our future work is concerned with the automatic extraction of domain models from the NL description. Such models will help analysts with the tedious work of developing domain models for complex software systems.

## REFERENCES

- Burg, J. F. M., van de Riet, R. P.: Analyzing Informal Requirements Specifications: A First Step towards Conceptual Modeling, Proc. of the 2nd Int. Workshop on Applications of Natural Language to Information Systems, Amsterdam, 1996.
- Fliedl, G., Kop, Ch., Mayerthaler, W., Mayr, H. C., Winkler, Ch.: The NIBA workflow: From textual requirements specifications to UML-schemata In: ICSSEA, Paris, 2002.
- Kop, Ch., Mayr, H. C.: Mapping Functional Requirements: From Natural Language to Conceptual Schemata, In Proc. of the 6th Int. Conf. SEA, Cambridge, USA, 2002.
- Lee, B.-S., Bryant, B.R.: Automated conversion from requirements documentation to an object-oriented formal specification language. In Proceedings of SAC(ACM), Madrid, Spain, 2002.
- Moreno, A.: Object-Oriented Analysis from Textual Specifications, In Proc. of 9th International Conference on Software Engineering and Knowledge Engineering (SEKE'97).
- Araújo, J., Moreira, A., Brito, I., Rashid, A.: Aspect-Oriented Requirements with UML. Workshop on Aspect-oriented Modeling with UML, UML 2002, Dresden, Germany
- Ilieva, M., Ormandjieva, O.: Automatic Transition of Natural Language Software Requirements Specification into Formal Representation, NLDB 2005.
- Ilieva, M., Ormandjieva, O.: Models Derived from Automatically Analyzed Textual User Requirements. Proc. of SERA'06
- Ilieva, M.: Graphical Notation for Natural Language and Knowledge Representation. In Proc. of 19th SEKE, 2007.
- Ilieva, M.: Use Case Paths Model Revealing Through Natural Language Requirements Analysis, Proceedings of ICAI, 2007.
- Ilieva, M, Ormandjieva, O.: NLP and FCA Technology for Automatic Building of DM, Proceedings of SEA, 2007.
- Abbott, R.: Program design by informal English descriptions, Communications of the ACM Volume 26/11 1983, pp. 882-894.
- Mihalcea, R., Liu, H, Lieberman, H.: NLP for NLP, CICLing 2006, LNCS 3878, pp. 319-330, 2006.

# Specification of data requirements from task descriptions

Jose Luis de la Vara and Juan Sánchez  
*Centro de Investigación en Métodos de Producción de Software*  
*Universidad Politécnica de Valencia*  
*Camino Vera s/n, 46022, Valencia, Spain*  
*{jdelavara, jsanchez}@pros.upv.es*

## Abstract

*When developing an information system for an organization, understanding of the application domain is essential for the requirements engineering process. The need of organizational modeling has been widely acknowledged, and business process modeling can be considered a must. The functional requirements of an information system, which can be considered the main type of requirement, must support business processes and can be elicited from them. Nonetheless, the behavioral perspective of an information system that functional requirements provide must be complemented with a data perspective. Both data and functional requirements must be specified, and means of helping system analysts to properly elicit and specify them are necessary so that the requirements are complete and consistent. This paper presents an approach that provides detailed methodological guidance to specify the data requirements of an information system from functional requirements. First, functional requirements are elicited from business process diagrams and are then specified in the form of task descriptions. Next, data requirements are specified by means of the information flows that the information system and its users exchange during the execution of the task descriptions. Information flows are specified on the basis of a BNF grammar by following a set of guidelines.*

**Keywords:** data requirement, functional requirement, information flow, task description, information system.

## 1. Introduction

Understanding of the application domain is essential for the requirements engineering (RE) process of a software system, so the need of organizational modeling for the development of an information system (IS) for an organization has been widely acknowledged (e.g. [2][7][20]). Business process modeling is part of most of the organizational modeling-based RE

approaches, and it can also be considered a must for IS development. ISs for organizations should manage and execute operational processes involving people, applications, and/or information sources on the basis of process models [9], and RE approaches for their development should differ from traditional ones [1]. First, detailed process models are necessary in the RE process. Second, new systems must support new ways of (better) running an organization.

In practice, inadequate functionality is the most common cause of software system failure to meet expectations [14]. As a result, the functional requirements of an IS usually receive most of the attention during the RE process and can be considered the main type of requirement. They must support the business processes of an organization, and business process diagrams (BPD) can be used for their elicitation. However, functional requirements are not sufficient to completely specify a system. The behavioral perspective of an IS that is provided by functional requirements must be complemented with a data perspective. Both functional requirements (which indicate what the system shall do) and data requirements (which indicate what the system shall store) must be taken into account.

Specification of functional and data requirements is a common recommendation for any RE approach [15]. They provide different system perspectives, thus they complement each other [22], and there exist several techniques and styles for their specifications [14]. However, problems may arise when specifying both types of system requirements. They are usually specified separately, and incompleteness and inconsistency between the specifications of data and functional requirements may appear if they are not properly managed [10][11][13]. Therefore, means of helping system analysts to properly elicit and specify them are necessary.

This paper presents an approach that provides detailed methodological guidance for the specification of the data requirements of an IS from functional requirements. The goals of the approach are: 1) to help



system analysts specify the data that will be stored and managed by an IS on the basis of its functional requirements; and, 2) to integrate the specification of functional and data requirements so that the problems described above do not arise.

The approach that is presented is the result of a collaborative project with the software development company CARE Technologies (<http://www.care-t.com>), and it is the extension of a wider RE approach [4][5][7] (referred to as business process-based approach hereafter) that encompasses business process-centred organizational modelling, system purpose analysis, business process reengineering (considered as improvement), and specification of functional requirements. CARE uses OO-Method [17], which is a methodology for automatic software generation based on data-centered conceptual modeling. Therefore, extending the business process-based approach with specification of data requirements is essential in order to properly integrate it in the software development process of the company.

Specification of data requirements from functional requirements is carried out as follows. First, functional requirements are elicited from BPDs and then specified in the form of task descriptions in a textual template. Next, data requirements are specified by means of the information flows that the IS and its users will exchange during the execution of the task descriptions. Information flows are specified on the basis of a BNF grammar by following a set of guidelines, and they are included in the textual template of task descriptions.

The rest of the paper is organized as follows: section 2 presents an overview of the approach; sections 3 and 4 describe the specification of task descriptions and of information flows, respectively; section 5 revises related work; finally, section 6 explains our conclusions and future work.

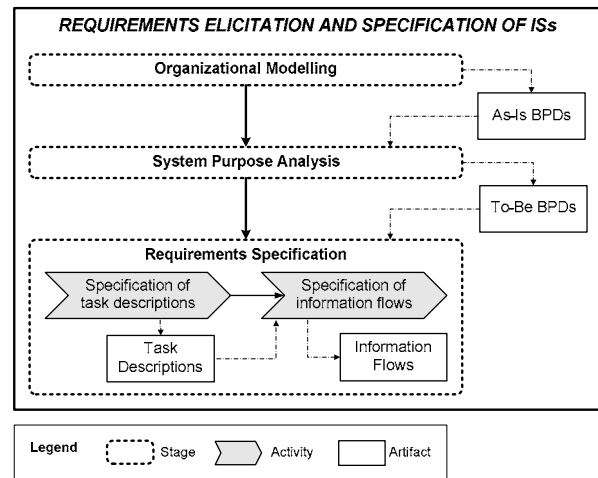
## 2. Approach overview

As a result of the need of integration in the software development process of CARE Technologies, it has been necessary to extend the business process-based approach so that it addresses data requirements.

For the business process-based approach, data requirements depict all the pieces of information that the task descriptions of an IS will need so that business tasks are properly supported. These pieces will be those that the IS and its user will exchange. Therefore, we advocate for the specification of the information flows between an IS and its user to specify data requirements. Completeness of data requirements will be reached if all the pieces of information that are

necessary for the execution of the task descriptions of an IS are specified, and consistency between data and functional requirements will exist if all the pieces of information that are specified are used as input or output of the task description

The approach that is presented in this paper addresses requirements specification, and it consists of two activities (Figure 1): specification of task descriptions and specification of information flows. Nonetheless, these activities could be regarded as a unique activity (specification of task descriptions and information flows). First, task descriptions are specified from to-be BPDs through the filling of a textual template, as explained in [5]. Next, information flows are specified from the content of the textual templates on the basis of a BNF grammar by following a set of guidelines.



**Figure 1. Approach for requirements elicitation and specification of information systems**

In summary, after extending the business process-based approach, the RE process of an IS for an organization that we propose is as follows (Figure 1):

**Stage 1. Organizational modeling:** the purpose is to understand the application domain by modeling the current organizational environment and to obtain the BPDs of the organization

**Stage 2. System purpose analysis:** the purpose is to understand and analyze the system goals and determine the effect on the business processes of the organization

**Stage 3. Requirements specification:** the purpose is to specify the functional requirements that will adequately support the business processes, and to specify the data requirements that represent the pieces of information that will be necessary for the execution of the functional requirements.

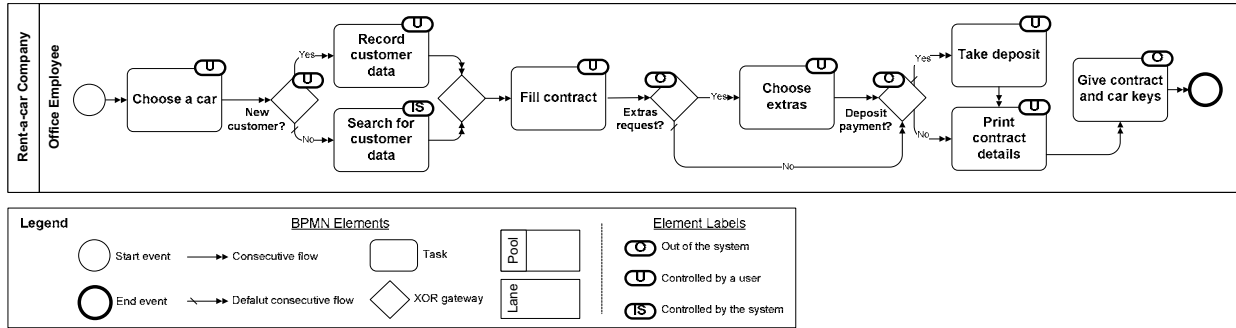


Figure 2. Business process “car rental” of a rent-a-car company

This paper focuses on the third stage of the business process-based approach. Organizational modeling and system purpose analysis are out of its scope. Details about them can be found in [4] and [7].

A very positive point of collaborating with CARE Technologies is that it belongs to a holding company (<http://www.olivanova.com>), so other organizations of this company have been used to evaluate the approach. These organizations are small/medium sized. As an example to explain the approach, a rent-a-car company (<http://www.rentacar-denia.com>) is used. Nonetheless, the complete case study is not explained and just some of the information is used.

Figure 2 shows the business process “car rental” modelled in BPMN [16]. Since the example is straightforward, the business process is not explained in great detail. Nonetheless, it must be pointed out that the BPD uses two elements that are graphical extensions to BPMN: labels (to represent the automation level of flow objects) and consecutive flows (to represent the fact that two flow objects are executed consecutively). Details about these elements can be found in [5].

### 3. Specification of task descriptions

The functional requirements of an IS are elicited from the to-be BPDs of an organization and specified by means of task descriptions in a textual template, as explained in [5].

Textual templates include the following information: the name of the task description; the subtasks and the business process that are supported; the role responsible for its execution; the triggers, preconditions and postconditions of the task description; the input and output domain entities and their states; the business rules that affect the task description; and the normal user intention and system responsibility (an abstract description of the interaction between a user and the IS) and their alternatives and

extensions. Guidelines to fill a textual template can be found in [5].

Figure 3 shows an example of task description that has been specified from the BPD shown in Figure 2. The domain entity “Customer” appears in input and output as Customer (1) and Customer (2) because they refer to different customers.

Task Description: CAR RENTAL			
<b>Business Process:</b> Car Rental		<b>Role:</b> Office Employee	
<b>Subtasks:</b> Choose a car, Check whether a customer is new or not, Record customer data, Search for customer data, Fill contract, Choose Extras, Take deposit, Print contract details			
<b>Triggers:</b> -			
<b>Preconditions:</b> -			
<b>Postconditions:</b> -			
Input		Output	
Domain Entity	State	Domain Entity	State
Car	Ready	Rental Contract	Open
Customer (1)	-	Customer (2)	-
Extra	Ready	Car	Rented
-	-	Extra	Rented
<b>Business Rules</b>			
• The insurance of a car must be valid during the rental period			
User intention		System responsibility	
<i>Normal</i>			
2. Select a car		1. Show cars	
4. Select a customer		3. Show customers	
5. Introduce rental contract information		6. Store information	
		7. Show contract details	
		8. Print contract details	
<i>Alternatives</i>			
4.a.1. Introduce customer data		4.a.2. Store customer data (→5)	
<i>Extensions</i>			
5.a.2. Select extras		5.a.1. Show extras	
5.b.1. Introduce deposit amount			

Figure 3. Example of task description

### 4. Specification of information flows

System analysts specify the pieces of information that an IS and its actors exchange for the execution of its task descriptions by means of information flows.

They are specified on the basis of the BNF grammar shown in Figure 4.

```

<Information flow> ::= <Input flow> |
<Output flow> | <Input flow> <Output flow>
<Input flow> ::= → <Data expression>
<Output flow> ::= ← <Data expression>
<Data expression> ::= <Domain entity> |
<Domain entity> / <Attribute> / |
<Data expression> + <Data expression> |
( <Data expression> ' | ' <Data expression> ) |
<Lower limit>{ <Data expression> }<Upper limit> |
[ <Data expression> ]
<Attribute> ::= <Attribute name> |
<Attribute> + <Attribute> |
( <Attribute> ' | ' <Attribute> ) | [ <Attribute> ]
<Domain entity> ::= <String>
<Attribute> ::= <String>
<String> ::= <Character> | <Character><String>
<Character> ::= <Letter> | <Digit> | _ |
<Lower limit> ::= <Digit> | <Digit><Digit>
<Upper limit> ::= <Digit> | <Digit><Digit>| n
<Letter> ::= A | a | B | b | C | c | D | d ...
<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

```

**Figure 4. BNF Grammar for the specification of information flows**

Information flows can be considered a specialization of data expressions [14], i.e. data expressions that are exchanged between an IS and its users during the execution of task descriptions. The main advantages of data expressions and, thus, of information flows are that are very compact, precise and easy for system analysts and stakeholders to use and understand.

Information flows were proposed by the info cases approach [10], and several principles for them are presented in [6]. However, these works do not provide a complete BNF grammar, do not explain the semantics on information flows in detail, and do not provide detailed guidance for the specification of information flows. In addition, the way of specifying information flows of the info cases approach is different from the way of the approach that is presented.

Information flows are an abstract representation of the communication between an IS and users. They are not as detailed as, for example, sequence diagrams or user interface prototypes. Furthermore, we think that these models are too solution-oriented to be adequate for requirements specification.

The input flow of an information flow represents all the pieces of information that a user has to communicate to (introduce in) an IS when executing a task description. The output flow represent all the pieces of information that an IS has to communicate to a user after executing a task description.

It is also important to point out that the semantics of the input and output flows of an information flow and the semantics of the input and output of a task

description are different. The input of a task description is the domain entities that exist and are used or consumed by its subtasks, whereas the output is the domain entities that are generated or changed after the execution of its subtasks.

The symbols that can appear in an information flow and their semantics are:

- ‘→’, for input pieces of information to an IS
- ‘←’, for output pieces of information from an IS
- ‘//’, for membership
- ‘+’, for aggregation
- ‘( | )’, for alternative
- ‘{ }’, for repetition
- ‘[ ]’, for option
- ‘n’, for indeterminate number of repetitions

It is essential that the information flows of the task descriptions of an IS allow system analysts to completely and consistently specify the pieces of information that depict the data requirements of the IS. These pieces of information will be those that are necessary for the execution of the task descriptions, and thus for the support of the business processes. In addition, precisely specifying the information flows from a BNF grammar allows the automation of the derivation of an analysis class diagram to be possible, although not completely, as described in [6].

A new section is included in the textual template of task descriptions to specify their information flows, so specifications of data and functional requirements are integrated. Specification of information flows is carried out from the domain entities that are used as input and output of task descriptions and from its interactions.

The guidelines to specify the information flow of a task description are presented in the following subsection. Nine guidelines have been defined, and they are grouped into three sets.

#### **4.1. Guidelines for the specification of information flows**

For each task description of an IS, an information flow is specified by following these guidelines:

##### **G1) Determination of necessary flows**

G1.1) An input flow has to be specified if there exist some action of user intention in which the user has to choose among the existing pieces of information in the system (domain entities) or has to introduce new pieces of information (domain entities and/or their attributes).

G1.2) An output flow has to be specified if the system has to report to users about existing pieces of information at the end of the task description (domain entities and their attributes).

## G2) Determination of pieces of information

(Normal)

- Car + Customer (1) + Rental Contract / contract number + current date + current time + office + return date + return office /
- ← Rental Contract / contract number + current date + current time + office + return date + return office + rental cost + extras cost + VAT + deposit + total cost / + Car / model + plate number / + Customer (1) / name + surname + ID number /

(Alternative 'a')

- Customer (2) / number + name + surname + ID number + address + city + telephone number + credit card type + credit card number + credit card expiration date /
- ← Customer (2) / name + surname + ID number /

(Extension 'a')

- of Extra }<sub>n</sub>
- ← of Extra / name / }<sub>n</sub>

(Extension 'b')

- Rental Contract / deposit /

## G3) Weaving

Task Description: <b>CAR RENTAL</b>
<b>Information flow</b>
→ <u>Car</u> + ( <u>Customer (1)</u>   <u>Customer (2)</u> / number + name + surname + ID number + address + city + telephone number + credit card type + credit card number + credit card expiration date / ) + <u>Rental Contract</u> / contract number + current date + current time + office + return date + return office + [ deposit ] / + [ ; { <u>Extra</u> } <sub>n</sub> ]
← <u>Rental Contract</u> / contract number + current date + current time + office + return date + return office + rental cost + extras cost + VAT + deposit + total cost / + <u>Car</u> / model + plate number / + ( <u>Customer (1)</u> / name + surname + ID number /   <u>Customer (2)</u> / name + surname + ID number / ) + [ ; { <u>Extra</u> / name / } <sub>n</sub> ]

Figure 5. Example of specification of information flows

## G2) Determination of pieces of information for each interaction

G2.1) For each interaction that could be carried out (normal, alternatives and extensions), the domain entities that are necessary in its flow(s) have to be determined. If there is more than a domain entity in a flow, then they are aggregated.

G2.2) For each domain entity that appears in an input flow, the pieces of information that represent attributes of the domain entities (obtained from stakeholders and organizational documentation) and that a user must introduce in the system have to be specified by means of memberships.

G2.3) For each domain entity that appears in an output flow, the pieces of information that represent attributes of the domain entity and that a user need to know after the execution of the task description have to be specified by means of memberships.

G2.4) For each data expression of a flow, its number of repetitions has to be specified.

## G3) Weaving of pieces of information

G3.1) The pieces of information of normal interaction that would not be exchanged between the

system and a user if an alternative was executed and the pieces of the alternative have to be weaved. The weaving is carried out by specifying alternatives in the flow(s) of the task descriptions.

G3.2) The pieces of information of extensions have to be weaved into the rest of pieces of information. The weaving is carried out by specifying options in the flow(s).

Figure 5 shows the outcome of applying the guidelines to the task description of Figure 3. The names of the domain entities are underlined in the flows, and the application of the first set of guidelines (G1) is not shown because its result is easy to understand (both an input flow and an output flow are specified).

## 5. Related work

The most common practice in the RE approaches that deal with functional and data requirements is to model classes from uses cases or jointly with them. They are based on mechanisms such as linguistic patterns [8], sequence diagrams [12], activity graphs [13] or consistency guidelines [11]. These approaches are solution-oriented, so they do not properly analyze the application domain, and they do not provide guidance for requirements elicitation. In addition, they are more complex than the approach that has been presented because they require the use of more models or models that are less flexible than information flows.

All the organizational modeling-based RE approaches that model business processes deal with functional and data requirements (e.g. EKD [2] and ARIS [20]). However, they lack precise guidance for requirements elicitation and specification and for assurance of consistency and completeness between data and functional requirements. Some approaches focus on data modeling from BPDs (e.g. [19]), but the models that are obtained are incomplete and guidance for completion is not provided. When compared with the business process-based approach, these approaches do not explain how to improve business processes.

Finally, several works have acknowledged the importance and benefits of a data-centered perspective when modeling business processes. They address issues such product-based workflow design of manufacturing processes [18], detection of data flow anomalies [21], and document-driven workflow systems [23]. These works take data into account from a perspective of data flow through tasks rather than from a perspective of information flows with an IS, and do not regard BPDs as a means for requirements elicitation.

## 6. Conclusions and future work

Organizational and business process modeling as a means for understanding the application domain are essential for IS development. BPDs and functional requirements play a major role for IS specification, but system analysts must not limit their focus on them alone. They must also take other aspects such as data requirements into account.

This paper has presented an approach that provides detailed methodological guidance to help system analysts specify the data requirements of an IS. Once the functional requirements of the IS have been specified in the form of task description, data requirements are specified by means of information flows between the system and its users when executing the task descriptions. A BNF grammar and guidelines have been presented to precisely specify the information flows and so that completeness of data requirements and consistency with functional requirements can be assured. In addition, the specifications of data and functional requirements have been integrated. The textual template for the specification of task descriptions has been extended with a new section to specify their information flows.

As future work, the approach must be applied in more projects in order to further evaluate it and so that improvements might be made. It is important that the new approach is used in large projects and in projects in which a legacy system exists. In addition, the development of tool support is planned to facilitate the use of the new approach and to automate the specification of information flows, and a technique for the analysis of system non-functional requirements and guidelines for the derivation of the presentation model (user interface) of OO-Method are necessary.

## 7. Acknowledgments

This work has been developed with the support of the Spanish Government under the project SESAMO TIN2007-62894 and the program FPU AP2006-02324, and co-financed by FEDER.

## 8. References

[1] I. Alexander, I. Bider, and G. Regev, "Workshop on Requirements Engineering for Business Process Support (REBPS'03), Objectives and Motivation", CAISE'03 Workshops, Klagenfurt/Velden, Austria  
[2] J. Bubenko, A. Persson, and J. Stirna, "EKD User Guide", 2001, [http://www.dsv.su.se/~js/ekd\\_user\\_guide.html](http://www.dsv.su.se/~js/ekd_user_guide.html)  
[3] L. Constantine and L. Lockwood, *Software for Use*, Addison-Wesley, Reading, 1999

[4] J.L. de la Vara and J. Sánchez, "Improving Requirements Analysis through Business Process Modelling: A Participative Approach", BIS 2008, Innsbruck, Austria  
[5] J.L. de la Vara and J. Sánchez, "BPMN-based Specification of Task Descriptions: Approach and Lessons Learnt", REFSQ'09, Amsterdam, Netherlands.  
[6] J.L. de la Vara, et al., "A Requirements Engineering Approach for Data Modelling of Process-Aware Information Systems", BIS 2009, Poznan, Poland  
[7] J.L. de la Vara, J. Sánchez, and Ó. Pastor, "Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems", CAiSE'08, Montpellier, France  
[8] I. Díaz, J. Sánchez, and A. Matteo, "Conceptual Modelling Based on Transformation Linguistic Patterns", ER2005, Klagenfurt, Austria  
[9] M. Dumas, W. van der Aalst, and A. ter Hofstede, *Process-Aware Information Systems*, Wiley, Chichester, 2005  
[10] M.H. Fortuna, C.M.L. Werner, and M.R.S Borges, "Info Cases: Integrating Use Cases and Domain Models", RE'08, Barcelona, Spain  
[11] M. Glinz, "A Lightweight Approach to Consistency of Scenarios and Class Models", ICRE'00, Schaumburg, USA  
[12] E. Insfran, Ó. Pastor, and R. Wieringa, "Requirements Engineering-Based Conceptual Modelling", *Requirements Engineering* 7(2), 61-72, 2002  
[13] G. Kösters, H.W. Six, and M. Winter, "Coupling Use Case and Class Models as a Means for Validation and Verification of Requirements Specifications", *Requirements Engineering* 6(1), 3-17, 2001  
[14] S. Lauesen, *Software Requirements: Styles and Techniques*, Addison-Wesley, London, 2002  
[15] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap", ICSE'00, Limerick, Ireland  
[16] OMG. Business Process Modeling Notation (BPMN) Specification v1.2, 2009, <http://www.bpmn.org>  
[17] O. Pastor and J.C. Molina, *Model-Driven Architecture in Practice*, Springer, Heidelberg, 2007  
[18] H.A. Reijers, S. Liman, and W.M.P. van der Aalst, "Product-Based Workflow Design", *Journal of Management Information Systems* 20(1), 229-262, 2003  
[19] A. Rodriguez, E. Fernández-Medina, and M. Piattini, "CIM to PIM Transformation: A Reality", CONFENIS 2007, Beijing, China  
[20] A.W. Scheer, *Aris - Business Process Modeling* (3rd edition), Springer, Heidelberg, 2000  
[21] S.X. Shun, J.L. Zhao, and J.F. Numaker, "Formulating the Data-Flow Perspective for Business Process Management", *Information Systems Research* 17(4), 374-391, 2006  
[22] K. Siau and L. Lee, "Are use case and class diagram complementary in requirements analysis?", *Requirements Engineering* 9(4), 229-237, 2004  
[23] J. Wang and A. Kumar, "A Framework for Document-Driven Workflow Systems", BPM 2005, Nancy, France

# From organizational models to software requirements

Alicia Martinez<sup>1,4</sup>

<sup>1</sup>Technical Institute of  
Zacatepec, Mexico.  
alimartin@dsic.upv.es

Oscar Pastor<sup>2</sup>

<sup>2</sup>Valencia University of  
Technology, Spain.  
opastor@dsic.upv.es

John Mylopoulos<sup>3</sup>

<sup>3</sup>University of Trento,  
Italy. jm@dit.unitn.it

Hugo Estrada<sup>4</sup>

<sup>4</sup>CENIDET, Mexico.  
hestrada@dsic.upv.es

## ABSTRACT

The early requirements phase, which is focused on the analysis of the business environment where a system will operate, is one of the current research priority areas for several research groups around the world. The late requirements phase, on the other hand, is focused on analyzing the expected functionality of the system-to-be. Late requirements analysis has been studied for years and is well understood. However, these requirements phases have generally been developed in isolation and we lack techniques that derive late requirements from early ones. The objective of this paper is to provide systematic guidelines to generate the requirements specification of the system-to-be from the relevant information of an organizational model. This work has been made in the context of OO-Method, a software production process that automatically generates complete systems from late requirements specifications.

## Keywords

Organizational requirements, software requirements.

## 1. INTRODUCTION

In recent years, many research efforts have been made to define software development processes to generate systems from software requirements. These approaches solve many of the issues associated with developing organizational software systems. However, they don't ensure that the system-to-be fits well its organizational environment.

Accordingly, many researchers working in the area agree that system requirements should be derived from an organizational model. McDermond [1] indicates that when the functional specification of the software system is the focal point of the requirements analysis, requirements engineers tend to establish the scope of the software system before having a clear understanding of user real needs. In this context, any attempt to generate a prototype of the information system will be reduced by the incapacity to assure beforehand the real usefulness of the system in the context of its organizational environment.

There are several research works that highlight the importance of using organizational models as a starting point in the development of information systems.

However, there is currently no an industrial software development environment that offers a methodological approach that is based on an organizational model for the generation of prototypes of information systems. The lack of traceability methods has affected the practical application of organizational model techniques in integrated software production process environments. Thus, we argue that the determination of a methodological approach to use the elements of an organizational model to obtain the expected functionality of the information system is a basic requirement to assure its usefulness in practice.

In this paper, we present a method to generate information system requirements from an organizational model represented in the Tropos Framework.

There are some works [2][3][4] that offer solutions to translate early requirements into software specifications (requirements and conceptual models respectively). One difference with our work is that we propose an intermediate model to reduce the abstraction level of the organizational model. The use of the proposed method in the context of an industrial project is also a difference with the other research works in the area.

The software requirements specification generated corresponds to a specific requirements approach RETO [5], which is the requirements method and tool associated to OO-Method. OO-Method is the CASE Tool that is being extended with an organizational modeling stage. The paper is structured as follows: Section 2 presents the foundations of the research work. Section 3 presents the overview of the proposal. Section 4 presents the method to extend the organizational model with concerned objects. Section 5 presents the generation of software requirements and finally, Section 6 presents the conclusions.

## 2. FOUNDATIONS

This section presents the methodologies used in this research work: The Tropos Framework and the OO-Method CASE tool. Both approaches are combined to obtain a requirements model of information systems.

### 2.1 The Tropos Methodology

Tropos [6] proposes a software development methodology and a development framework which are based on concepts used to model early requirements. They are based on the premise that in order to build software that

operates within a dynamic environment, it is necessary to analyze and explicitly model that environment in terms of actors, their goals and dependencies on other actors.

To support modeling and analysis during the early requirements, Tropos adopts the concepts offered by *i\** [7], a modeling framework defined in terms of concepts such as actors and social dependencies among actors, including goal, softgoal, task and resource dependencies. In Tropos we have the following key concepts: a) Actor: An actor is an active entity that carries out actions to achieve goals by exercising its know-how. b) Dependency: A dependency describes an intentional relationship between two actors: the *dependor* and the *dependee* that wait for a *dependum*. There are four types of dependencies: goal, resource, plan and softgoal dependencies. By using these elements, it is possible to define the Tropos models: a) The *Actor Model* shows the dependencies that exist between the organizational actors to achieve their goals, carry out tasks and provide or request resources, b) The *Goal Model* represents the tasks that have to be carried out by the actors to achieve the goals which are expected of them. This model considers means-end and decomposition links.

The Tropos and *i\** Frameworks have been used in several application areas, including requirements engineering, agent-based software generation, security modeling, business process reengineering etc. However, in Tropos methodology, there are still no methods to use the organizational models to produce object-oriented information systems in an automatic way within an industrial software production context.

## 2.2 OO-Method CASE Tool

The research work presented in this paper has been made in the context of the OO-Method project. OO-Method is an industrial, model-transformation method that relies on a CASE tool [8] to automatically generate complete information systems from software requirements models. The OO-Method can be viewed as a method where the focus is place on properly capturing system requirements in order to manage the complete software production process. The conceptual model, which is semi-automatically generated from a software requirement model, specifies the problem to be solved (problem space). Then, an abstract execution model is provided to guide the implementation of these requirements in a specific software development environment (solution space).

The implementation of the corresponding set of mappings between conceptual constructs and software representations constitutes the core of a Conceptual Model Compiler. The final software product is functionally equivalent to the requirements specification.

However, at the present time OO-Method does not have mechanisms to ensure that we are capturing the correct

requirements for the system-to-be. To do this, we need to include an organizational modeling stage as first phase of the OO-Method software production process.

## 3. OVERVIEW OF THE PROPOSAL

It is important to point out that one of the main objectives of this paper is to define a systematic approach to generate late requirements specifications that correctly fit the objectives of the organizational actors. To do that, the proposed method starts with the definition of an organizational model that represents the relevant actors and their goals (Fig. 1). Following, a goal analysis process is carried out in order to identify the relevant plans that fulfill the organizational goals (process 1). As result of this process, the relevant plans to be automated are identified. In process 2 we use a pattern language to generate a new organizational model where the software system is represented within its operational environment besides its functions and relevant characteristics. Transformational rules between models are used to ensure the traceability that is needed in the model-transformation approach of OO-Method. These initial processes of the proposed method were previously analyzed in [9].

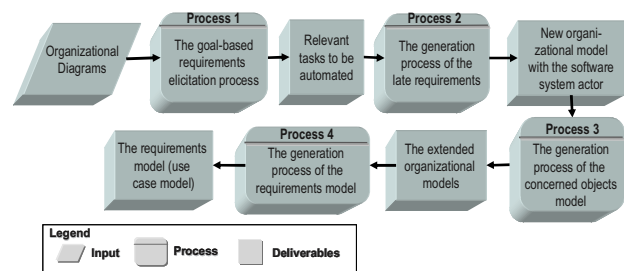


Fig. 1. Overview of the proposed method

In this paper, the analysis begins by understanding the organizational context, where the software system actor (SSA) has been inserted in the organizational model. Thus, the process 3 consists in to extend the organizational model with relevant objects, which we called concerned objects. The result of these stages is the input for the generation process of the requirements model which is explained as a contribution of this paper (process 4). All these previous phases were developed using extensions to Tropos.

The evaluation of the methodological approach proposed in this paper has been done using several industrial cases studies [10]. In this paper we present the *Car Rental Management* project, which concerns modeling the basic process for a real car rental enterprise in Alicante, Spain.

## 4. EXTENDING THE ORGANIZATIONAL MODEL WITH CONCERNED OBJECTS

This section describes the process to extend the organizational model to identify the relevant information in the definition of the system-to-be.

#### 4.1 The Concerned Object Model

In order to reduce the abstraction level of the organizational model, a concerned object model has been proposed (as an intermediate model between the early and late requirements models) in order to represent all the relevant information to be considered in the definition of the system-to-be.

A concern expresses a specific interest in some topic pertaining to a particular system of interest (or other subject matter) [11]. It is important to point out that concerns do not exist until someone is *concerned* about them. For example, in our proposed method, a business plan does not constitute a concern until an analyst has some reason to be interested in a plan as a candidate for functionalities in the system-to-be.

We use the concept of *concerned object* to represent an entity of interest in the process of defining the system-to-be. Therefore, the concerned objects extend the organizational model to facilitate the generation of software requirements. A *concerned object* represents a resource that is used within the organizational process, or an abstract entity that will be used in the system-to-be. The concerned objects sources are plans, resources and goals. Fig. 2 shows an example of a *concerned object* with its set of attributes.

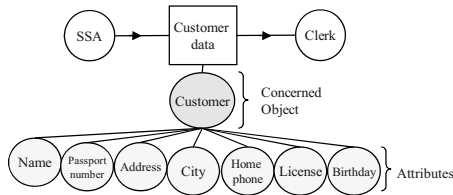


Fig. 2. Primitives of the concerned object model

#### 4.2 Rules for identifying concerned objects

A reduced version of the rules to create the concerned object model is presented below.

**Rule 1:** A resource dependency between the SSA (software system actor) and another organizational actor can be extended with one or several concerned objects.

**Rule 2:** The attributes of the resource will be the attributes of the created concerned object.

**Rule 3:** A plan executed in the organizational context can be extended with one or more concerned objects.

**Rule 3.1** When a plan uses or modifies a resource, the plan must be extended with a concerned object that represents the resource.

**Rule 3.2** If a plan uses or modifies a resource that has not yet been identified as a concerned object, then the plan must be extended using this resource to create a concerned object.

**Rule 3.3** When a plan does not use or modify any resource, then the plan does not need to be extended with a concerned object.

**Rule 3.4** A *composite plan*<sup>1</sup> needs to be extended with the concerned objects that include its children nodes. For example, Fig. 3 shows the structure of a *composite plan* and its associated *subplans*, where the concerned objects identified in the *subplans* are used to define the concerned objects of the *composite plan*.

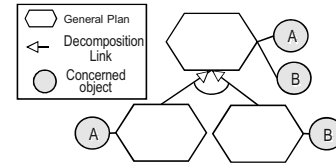


Fig. 3. Example for extending a composite plan

**Rule 4:** The characteristics of the resources used in the execution of a plan must be used in the identification of the associated attributes of the concerned object identified.

**Rule 5.** A *hardgoal* is a candidate to be extended with concerned objects if the goal is involved in a *means-end link* where the children nodes are plans that are associated with concerned objects.

The concerned object model will be the basis for the generation of the requirements model for the system-to-be.

### 5. LINKING LATE REQUIREMENTS WITH THE OO-METHOD REQUIREMENTS MODEL

This section describes our method to generate a requirements model from organizational models represented in the Tropos Framework.

The analysis performed in previous steps (business goals analysis, extension of the organizational model with the software system actor and extension of the organizational model with the concerned objects) are the basis to obtain the appropriate information to generate the requirements model. The process to discover the use case model is carried out by doing the following: a) Defining functional groups, b) Discovering default use cases, c) Discovering use cases through the analysis of the SSA, d) Discovering use case actors, e) Discovering relationships between use cases, and f) Building scenarios for use cases.

#### 5.1 Defining functional groups

A functional group describes the different subsystems that an information system can be divided into. Each functional group makes reference to an element that is manipulated (through user's interactions) by the software system.

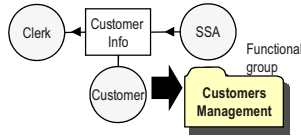
In our approach, the source model to obtain the functional groups is the actor diagram that has been extended with the concerned objects. In this model, the dependencies that associate an organizational actor and the SSA will generate the functional groups. To do this, Rule 1 and Rule 2 need to be applied.

<sup>1</sup> Those elements whose execution is carried out by decomposing them into other sub-elements.



**Rule 1.** Each concerned object identified in a resource or plan dependency between an organizational actor and the SSA must be mapped to a functional group.

**Rule 2.** The name of the functional group is composed of the name of the concerned object and the word “Management”. Fig. 4 illustrates an example of the creation of a functional group from an actor dependency.



**Fig. 4. Customer Management functional group**

### 5.2 Discovering use cases by default for each functional group

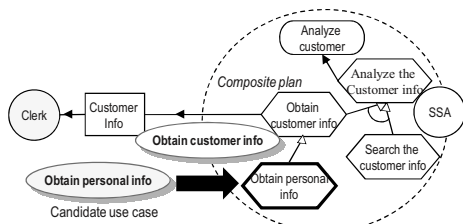
Along the development of case studies for this research work we found that a set of basic use cases must be defined in each functional group to manage the analyzed informational resource (create, delete and modify elements). These default use cases must be inserted in each functional group to ensure the correct management of the analyzed requirements.

**Rule 3.** Default use cases *Create*, *Delete* and *Modify* must be created for each functional group elicited in the previous steps. These use cases allow us to ensure the appropriate management of each functional group. The functional groups for the running example are: *Create Customers*, *Delete Customers* and *Modify Customers*.

### 5.3 Discovering use cases through the analysis of the SSA

The next step consists of determining the use cases from the organizational model that was extended with the inclusion of the software system actor. Therefore, an analysis of the internal element of the SSA must be carried out to determine its relevance in defining use cases. The rules associated with this step are the following:

**Rule 4.** Each plan within the SSA that is directly involved in a dependency relationship will be a candidate to be a use case in the requirements model. For example, Fig. 5 shows the plan *Obtain Customer info*, which is involved in a dependency relationship between the SSA and the Clerk actor. Therefore, it can be considered as a candidate to be a use case.



**Fig. 5. Example of a use case generated from an internal plan.**

**Rule 5.** Each plan within the SSA that is not involved in a dependency relationship could be a candidate to be a part of another use case in the requirements model

**Rule 6.** If the plan or goal (which has generated a use case) is linked to a dependency relationship, then it is necessary to determine if this use case must be contained in the functional group created from the dependency relationship.

Before allocating the use case in the functional group, it is necessary to analyze if the candidate use case corresponds to the semantics of some use case created by default in the functional group (*Create*, *Destroy* or *Modify*). If so, the candidate use case must substitute the use case created by default.

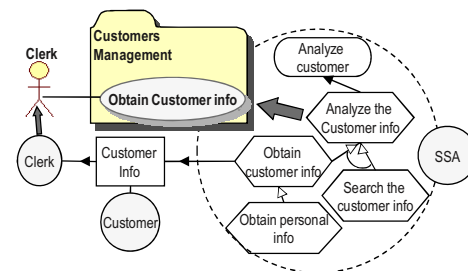
### 5.4 Discovering use case actors

*Actors* are parties outside the system that interact with the system [12]. In this proposal, the identification of actors is carried out by analyzing organizational actors and the roles or agents in the business, which have some kind of interaction with the SSA. Rule 7 defines the actor generation process.

**Rule 7.** The organizational actors with a dependency relationship with the SSA will be candidates to be actors of the requirements model.

**Rule 8.** Plans without a direct association to dependency relationships do not generate actors.

Fig. 6 shows an example of the application of Rule 7 to discover an actor of a use case. The plan *Obtain Customer info* has generated a use case with the same name as the plan. The dependency relationship associated to the plan is analyzed to determine the actor that participates in the dependency (Clerk). As a result of applying Rule 7, this organizational actor is translated into the actor that activates the use case *Obtain Customer info*.



**Fig. 6. Example for discovering an actor of a use case**

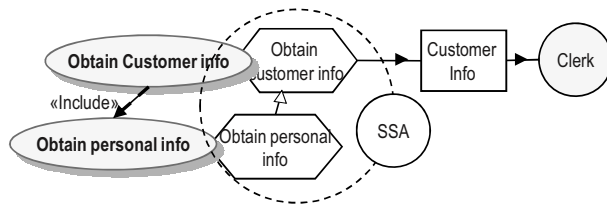
### 5.1 Discovering relationships between use cases

The fourth step of the process to generate the use case model consists in discovering the relationships between use cases. The UML standard supports three major relationships among use cases: include, extend and generalization; they can be summarized as follows [12]:

**Rule 7.** An *«include»* relationship must be created between use cases when the *composite* plan in a

composition plan relationship has generated a use case and its associated subplans have also generated use cases (applying Rule 5). Therefore, an «include» relationship between these use cases must be created, where the use cases generated from *subplans* are included in the use case generated from the *composite* plan.

Fig. 7 illustrates a partial view of the *Car Rental Management* case study. In this example, the application of Rule 4 to the *composite* plan *Obtain Customer info* generates a use case. A use case was also generated for the child node *Obtain personal info* through the application of Rule 5. Therefore, an «include» relationship between these use cases is created.



**Fig. 7 Example of the «include» relationship in the Car Rental Management case study**

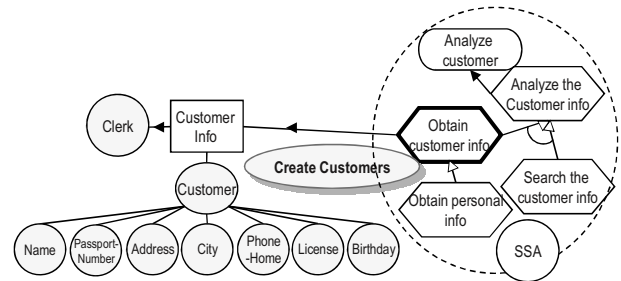
**Rule 8.** An «extend» relationship must be placed between two use cases when a plan need to be monitored by a special (monitoring) plan. Thus, this plan generates a use case and the monitoring plan also generates a use case.

**Rule 9.** The Tropos framework includes modeling primitives to represent agents, roles and positions. In our proposal, the concept of role is used to generate the generalization relationships.

## 5.2 Guidelines to obtain use case scenarios from organizational models

In this proposal, the process to build scenarios begins by selecting one of the elicited use cases. Then, the plan which was the source for that use case must be analyzed in order to obtain the use case scenario. In this case, the resource relationships associated to the plan that generates a use case are also a correct source for the generation of use case scenarios. The following rule helps in the construction of use case scenarios.

**Rule 11.** The resource relationships permit the functional groups to be determined; they also help to deduce the steps of the scenario. For example, Fig. 8 shows the resource dependency *Customer info*, where the SSA depends of the *Clerk* actor for obtaining the *Customer* information (i.e., *Name*, *Passport-Number*, etc). In this way, some steps for the *Create Customer* use case can be deduced. For example, (1) the system requests the *Customer* information; (2) the clerk introduces the *Customer* information, etc.



**Fig. 8. Resource relationship to obtain some steps for the create customer case study**

The following guidelines were developed to help the analyst in the process of obtaining use case scenarios from organizational models.

**Use case name:** The use case name in the template of the scenarios will be the same as the use case elicited using Rule 3, Rule 4, Rule 5, and Rule 6, where the use cases were determined.

**Use case actors:** The actor (s) of the use cases will be those actors that interact with the SSA through dependency relationships which were source of use cases (Rule7 and 8).

**Use case pre-Conditions:** The preconditions for the use case will be the same as the precondition of the plan which generates the use case.

**Use case purpose:** The explanation about the purpose of the use case must be written by software engineers based on the goals that operationalize the plans used to generate the use case model. These goals represent the rationalities behind the plans of the SSA.

**Use case relationships:** The relationships *include* and *extend* must be specified according to the relationships generated among internal plans in the SSA (Rule 8, Rule 8 and Rule 9).

**Use case basic course of action:** This information will be obtained by analyzing the elements associated to the plans that were the source of the generated use case. In the case of Tropos decomposition, it implies that the fulfillment of the child nodes implies the fulfillment of the parent node. Therefore, we can argue that these internal refinement structures will be the basis to define the actions associated with a use case. It is important to point out that one of the aspects that can not be obtained of the Tropos model is the temporally ordered actions that define the flow of the use case. This is because Tropos is not well-equipped to represent the execution order or the business plans. Therefore, we need the analyst intervention to order the actions involved in the use case.

At this point, it is important to identify those resources or plans where the actors that are associated to the SSA play the role of *dependee* in the dependency relationship (i.e., the system waits for actions or resources of the organizational actor) because the actions associated to this

dependency must be used to indicate the user intervention (column *actor communications* in the scenario template [5]). On the other hand, those resources or plans where the actors that are associated to the SSA play the role of *dependor* in the dependency relationship (i.e., the organizational actors wait for actions or resources of the system) must be analyzed to specify the system responsibilities (column *system response* in the scenario template). Rule 11 must be used to specify this situation.

An example of the use case model generated by applying our proposed rules to the running example is shown in Fig. 9, where the use cases of the *Customer Management*, *Reservations Management*, and *Cars Management* functional groups are shown.

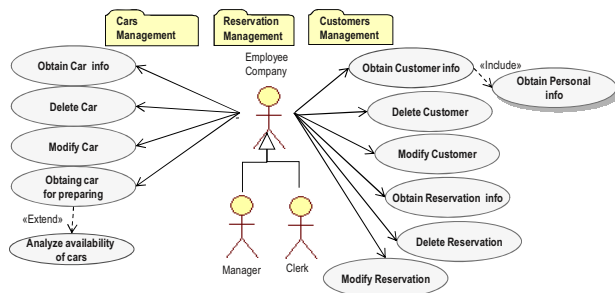


Fig. 9 Use Cases of the case study

## 6. CONCLUSIONS

We have proposed a set of guidelines that establish a correspondence between the modeling elements of an organizational model and those of a requirements model for the system-to-be. The guidelines help the analyst to define the system functionalities from the business plans. To do this, several steps must be fulfilled to generate an organizational model that integrates the software system as an explicit actor in the model. Several steps have been previously defined to reduce the abstraction level of a “pure” organizational model based on a pattern language.

As a result of these previous analyses, an organizational model that includes the software system actor (SSA) is created. The organizational model that is extended with the SSA is the basis to generate the requirements for the system-to-be.

The use of an intermediate model (organizational model with the SSA) is one of the differences of the proposed method with current research works in the area, where the software requirements are directly generated from organizational functionalities.

It is important to point out that the generation of the requirements model is a simple process based on model transformation rules. This transformation is possible because the level of the intermediate model is closer to the requirements model. In this sense, the intermediate model represents the expected functionalities of the system-to-be. This is also the kind of information that is represented in a

UML requirements model. Thus, we consider that both models represent the same information but represented in a completely different manner, one using UML and the other using an extension of the Tropos Framework.

## REFERENCES

- [1] McDermid J., *Software Engineer's Reference Book*. Butterworth-Heinemann Ltd. USA. 1991.
- [2] Santander F.A.V., Castro J., *Deriving use cases from Organizational Modeling*. In proceedings of the 10<sup>th</sup> International Conference on Requirements Engineering, pp. 32-39. University of Essen, Germany, 2002.
- [3] Castro J., Mylopoulos J., *Integrating Organizational Requirements and Object Oriented Modeling*. In proceedings of the 5<sup>th</sup> International Symposium on Requirements Engineering, pp. 146-153. Toronto, Canada. 2001.
- [4] Ortín M. J., García M. J., Moros B., Nicolás J. *El modelo de Negocios como base del Modelo de Requisitos: utilizando UML*. Jornadas de Ingeniería de Requisitos Aplicada, Sevilla, Spain, June, 2001.
- [5] Insfran E. *A Requirements Engineering Approach for Object-Oriented Conceptual Modeling*, PhD Thesis, Department of Information Systems and Computation, Valencia University of Technology, Spain. 2003.
- [6] Castro J., Kolp M., Mylopoulos J. *Towards Requirements-Driven Information Systems Engineering: The Tropos Project*. Information Systems 27(2): 365-389, Elsevier 2002.
- [7] Yu E., *Modelling Strategic Relationships for Process Reengineering*, PhD Thesis, University of Toronto, Toronto, Canada, 1995.
- [8] Pastor O., Gómez J., Infrán E., and Pelechano V. *The OO-Method approach for information systems modeling: from object-oriented conceptual modeling to automated programming*. Information Systems, 26(7): 507-534, Elsevier. 2001.
- [9] Martínez A., Pastor O., Mylopoulos J., Giorgini P., *From Early Requirements to Late Requirements: A goal-based approach*, in proceedings of the Eight International Bi-Conference Workshop on Agent-Oriented Information System (AOIS-2006), Luxembourg, Luxembourg, June, 2006.
- [10] Martínez A. *Conceptual Schemas Generation from Organizational models in an Automatic Software Production Process*. PhD. Thesis, Valencia University of Technology, Valencia, Spain, 2008.
- [11] Hilliard R. *Aspects, Concerns, Subjects, Views...\**. In proceedings of the First Workshop on Multi-dimensional Separation of Concerns in Object-oriented Systems at OOPSLA. Denver, USA.1999.
- [12] *UML Specification*. V1.3 Alpha R5, March 1999. Obtained from <http://www.rational.com/uml/index.jhtml>.
- [13] *OMG's Issue Reporting Procedure. Unified Modeling Language: Superstructure*, version 2.1.1. February 2007. Obtained from: <http://www.omg.org/docs/formal/07-02-03.pdf>.

# Systematic Review of Requirements Reuse

Flávia Braga de Azambuja

Universidade Federal de Pelotas (UFPel)  
Instituto de Física e Matemática  
Pelotas, RS, Brazil  
azambuja@ufpel.edu.br

Ricardo Melo Bastos, Ana Paula Terra Bacelo

Pontifícia Universidade Católica do  
Rio Grande do Sul (PUC-RS)  
Faculdade de Informática  
Porto Alegre, RS, Brazil  
{bastos, ana.bacelo@puccrs.br}

## Abstract

*Requirements Engineering is a field of study in software engineering that has been highlighted as a necessary task for an effective development process. Requirements reuse may be a new alternative to make engineering requirements tasks more systematic. This paper presents the results of a systematic review about the state of the art in requirements reuse. Systematic review is a research methodology that uses systematic methods to identify, select, and critically evaluate scientific studies in a specific field of research. The results of this work show that there is a large diversity of techniques and strategies for modeling of requirements. This diversity raises difficulties for the process of reuse in a systematic way requiring research efforts in this subject.*

## 1. Introduction

In the last decades, many techniques have been developed to support software reuse based on the premise that systems related to a same application domain presents similarities and as a consequence offer potential of reuse. Market demands related to time-to-market and quality of products, as well as the great diversity of platforms and existing languages, underlies the establishment of reuse practices as a way to reduce development time and decrease costs. Most researches in software reuse consider only reuse of source code and software components. However, some researches has considered the reuse in the others phases of the life cycle in a systematic manner. Based on these premises, a study was carried out to determine the state of the art of Requirements Reuse in order to identify the main gaps and challenges of research in this field. To reach this goal a method-

ology of systematic review was adopted. Systematic review is a research practice, often used in the medical field, which has been recently applied to the software engineering field. This methodology was adapted to software engineering by [12] and [4], in which the authors specify specific guidelines. This process helps to establish scientific rigor which is necessary to define the state of the art and to produce more reliable results. This paper is organized as follows: Section 2 starts with an overview of requirements engineering and requirements reuse to establish a background in this field. Section 3 presents the Systematic Review research methodology applied in this work. Section 4 describes the plan of systematic review and in section 5 and 6 the results and discussions of the systematic review are presented. Finally, in section 7 the findings and suggestions for future works are described.

## 2. Requirements Reuse

Requirements reuse is related to the concept of software reuse. Although it was established many years ago by [17], it has only been recently recognized as a common practice within the process of development. Reuse can be found not only in requirements phase, but also in all phases of software development life cycle (e.g components, artifacts, pre-existing knowledge). The requirements reuse is considered very incipient in practice requiring efforts to define requirements models and methods in order to improve its applicability by software industry. Systematic reuse needs operational support to its execution. In [15] the author highlighted the necessity of establishing an efficient form of representing and storing the specifications to allow comparison, adaptation, and management of the reusable elements, which can be reused during the whole development process. Currently, there is a large amount of modeling techniques

for requirements. Nevertheless, it is necessary to identify which approaches or processes are being applied, so as to establish a norm/standard that will ensure the reusability of the requirements when applied in all phases of the life cycle. The study of state of the art helps us to analyze how reuse of requirements has been proposed and applied. Considering the researches in Requirements Engineering field, [27] proposed a classification that establishes two research dimensions: researches addressed to problems and researches addressed to solutions. Although this approach was defined some time ago, it can be considered very current since many research projects still work on the investigation of solutions for these problems indicated by [27]. Inside the first dimension there are three classifications addressing Requirements Engineering problems: 1. The first classification explains the problems related to goals investigation, functions and obstacles of systems engineering which are done during the requirements and analysis tasks; 2. The second one includes the problems of behavioral specification of the system. This classification addresses problems related to information synthesis and the choice among alternatives to create a precise and minimal software specification. 3. In the third one, the problems are classified to establish how to reuse the requirements in other phases of the software development process. In [21] a classification that considers the main fields of research in Requirements Engineering is proposed. The author pointed out some research subjects that have not been solved yet. These subjects are similar to problems presented by [27]. Considering the problems highlighted by [21], the authors discussed the need to establish efficient methods of requirements model reuse. Concerning the works of [27] defined some concepts referring to the five types of tasks of Requirements Engineering and their main challenges. Once again, the problem of reuse was identified amongst the research fields and it should be investigated by this research area. Besides the tasks presented above, [1] proposed nine research hot spots, out of which six emerged from the future needs of software. They have been established due to the increase in scalability, security and dependence between the software and its environment. The other three hot spots have focused on the extension and maturity of the existing technologies for the improvement of Requirements Engineering methodologies and for the reuse of requirements. Thus, it is vital to find solutions for reuse field. The reuse of requirements brings more agility during their specifications since they are based on previous projects of similar products. However, in a practical context, one of the remained problems is how to identify real situations of Requirements Reuse. This is largely due to fact that part of reuse has been done by programmers in an informal manner. In some cases, experience helps the developers to reuse (e.g. source code, patterns, frameworks) because there are a lot of features in common among many applications in the same

domain [19]. Although this informal reuse shows evidence of advantages of Requirements Reuse models, it does not provide a standard form to systematically reuse the requirements. With the objective of determining the state of the art in Requirements Reuse and establishing the main gaps and challenges of the research in this field, the methodology of systematic review was applied as described in section 3.

### 3. Systematic Review in Software Engineering

Systematic Review is a research methodology that uses systematic methods to identify, select, and critically evaluate scientific studies in a specific field of research. It is a planned review to answer a specific question that can or not include statistic methods. Statistic methods used in the analysis and synthesis of the selected studies are called meta analysis [11]. In this work, the meta-analysis stage was not adopted since it is a qualitative diagnosis of studies of the field being researched. Differently of other fields of study, Software Engineering has some specificities that make it more difficult to obtain evidence through systematic review [4]. However, the use of systematic review methodology can be very useful to delimitate new researches. The systematic review, give a scientific rigor to a literature review process and, as a consequence, minimize the slants that can happen during a conventional literature review. The guidelines to lead the process of systematic review established by [12] and [4] were adapted to reflect the specific problems of research in Software Engineering. These guidelines are composed by three stages: planning of the review; conducting the review; and reporting the review.

### 4. Systematic Review Plan

The first stage in a systematic review begins with the definition of the research question. In this work we have adopted the term Question Focus (QF) as a way to represent the research question. The QF is essential to determine the structure of the review. If the QF is not well defined, it could substantially compromise the result of the research. To determine the QF some complementary issues were used. The subsections 4.1 and 4.2 summarize the sequence of steps that were established by [12] and [4]. In the section 5 the third stage is represented through de results analysis.

#### 4.1. Research Scope

Considering the general scope of the research, the goal is to find solutions for the following questions: ( $Q_x$ ):

- $Q_1$ : Which are the existent approaches and solutions related to requirements?

- *Q2*: How is Requirements Engineering prepared for systematic reuse?

- *Q3*: Which are the methodologies applied during the process of Requirements Engineering that support reuse?

These questions were used to delimitate the scope that will effectively be answered by the process of systematic review established in this work, through the analysis and synthesis of the selected studies. Based on the pre-identified questions, the *QF* was defined as follows:

- *QF*: “Which are the methodologies, strategies or advices being used in Requirements Engineering that support reuse?”

All the steps of the systematic review (i.e. project development, identification and selection of studies, data extraction, quality evaluation, analysis, presentation, and interpretation of the results) were guided by the *QF*, which was also used as a way of judgment of the systematic review relevance.

#### 4.2. Systematic Review Details and Protocols

This systematic review uses the following search engines: IEEE Xplorer digital library, ACM digital library, Springer Link and Science Direct. The criteria to make the decision about the selected search engine were: 1. the database allow search engines based on key words and Boolean expressions; and 2. the availability of articles through Internet. The selection of papers occurred in 2 months. The population defined for the study includes articles published in journals and conferences on the field of study since 2004 until now and that were written in English language. The period was delimited due to the need of establishing in this study the state of the art in research on Requirements Reuse. The key words used to do the search of articles are: “requirements reuse”, “requirements engineering”, “reuse” and “systematic reuse”. These key words were combined through Boolean operators and filters, as presented in Table 1. Considering the articles obtained by the filters, it is only considered those that refer to the following fields: computer science, software engineering, information systems, requirements engineering, product line; and artificial intelligence. The sources were exclusively accessed on the web, so manual search was not considered in the context of this work.

ID	Search String
S1	“((requirements reuse)) <and> (pyr >= 2004 <and> pyr <= 2008)”
S2	('requirements engineering' <and> reuse)) <and> (pyr >= 2004 <and> pyr <= 2008)
S3	('requirements engineering' <and> 'requirements reuse')) <and> (pyr >= 2004 <and> pyr <= 2008)
S4	('requirements engineering' <and> 'systematic reuse')) <and> (pyr >= 2004 <and> pyr <= 2008)

**Table 1: Search Strings.**

The 102 articles selected were resulting of the string S1. The string selects articles from journals and conference published since 2004 which deal with Requirements Reuse. It is important to emphasize that the S1 string was chosen for includes a larger number of works that deal the subject, providing more useful results.

### 5. Results

The 102 articles were preliminary selected for a future analysis. Hence, this selection was based on the following criteria: description of the approaches, methods, strategies and tools to solve the problem of research related to requirements reuse. After this selection, 84 articles were discarded. The selected articles were classified according to the criteria adapted from [1], [26] and [22]. The analysis of the papers consists specially of contributions in reuse field. The authors proposed to group the studies in three categories of solution technologies: a) Methodologies, Strategies or Advice; b) Kinds of Solution and c) Application. The categories are presented below:

- a) Methodologies, strategies or advice: this category identifies the methodologies, strategies or advice adopted in the selected studies focusing on reuse;
- b) Kinds of Solution: this category indicates the kind of solution adopted by each study with the focus on reuse. In addition, the studies of this category were classified according [26] to:
  - Problem investigation: investigates the current situation;
  - Solution design: proposes an improvement to the current situation;
  - Solution validation: investigates the proposed solution properties;
  - Solution selection: shows the improvement among the proposals from the literature;

- Solution implementation: realizes the solution;
  - Implementation evaluation: investigates the new situation, for example, when is investigated the practice of Requirements Engineering in an organization, where this organization has recently introduced a new way of doing Requirements Engineering.
- c) Application: in this category the studies were organized as proposed by [22]:
- Survey: focuses on obtaining the same kinds of data from a large group of people (or events), in a standardized and a systematic way;
  - Design and creation: focuses on developing new products or artifacts;
  - Experiment: focuses on investigating cause and effect relationships, testing hypothesis and seeking to validate a casual link or not between a factor and an observed outcome;
  - Case study: focuses on an instance of the thing that is to be investigated;
  - Action research - focuses on research in a real situation;
  - Literature Review - focuses on synthesizing results into a summary of what is and is not known; on identifying areas of controversy in the literature; and on formulating questions which need further research.

The classification of the studies based on these criteria is presented in Table 2, and the discussion in section 6.

## 6. Discussion

The correct elicitation, comprehension, and representation of the requirements are critical steps in the development of systems with focus on reuse. This is a highly cognitive activity and its success or failure depends on the skills and previous experiences of the involved requirements engineering team. The current methods and tools still do not give the expected support to this activity. One way to make it more systematic is facilitating reuse through the pre-existing requirements artifacts (e.g. models, specifications and so on). Product line has been considered as one of the most used strategy to obtain requirements through the reuse. The products are part of a product family and the dependencies among them have been planned since the beginning. There are some works that present issues related to both, Requirements Reuse and Product Line [5],[6],[7] [8], [20] and [13]. Software product lines may guide comprehensively the construction of all artifacts produced during

the system development for reuse. These classical product line approaches provide mechanisms to handle requirements for reuse [13]. The main challenges for Requirements Engineering in product lines include effective strategies and techniques for the analysis of domains as well as the documentation of requirements. In [1] the author highlighted some techniques and strategies that have been applied in this field using feature models, multi-agent feature trees, pattern modeling, and problem frames. Feature models are commonly used to model the characteristics of the domain, but in [1] the authors discuss that features reproduce themselves too fast when used to model instantiations of a specific domain. We also observe that features modeling activity is more related to requirements elicitation activity than requirements specification ones. However, it is important to have good specification of these features to be more consistent and to support the creation of other domain artifacts and the creation of product from features models. Multi-agent feature trees appear as a new promise for solutions, but have not been investigated enough. Another form of reuse is through modeling of patterns in which usable structure models are codified. Some works as [10], [20] suggest the codification of patterns to improve accuracy and completeness of the requirements. As a consequence, it can reduce the time to produce a specification due to the fact that the requirements are already known from other existent projects. So, a pattern is considered an abstraction that requires few adaptations in a project related to the same domain or product family. Reusable requirements could be composed by standard fields, such as the context of the problem, consequences, properties, among others [18]. However, this information is still not enough to facilitate the effective use of patterns or reusable artifacts. The adequate adaptation and instantiation of a pattern, so as to adjust the desired context, is still considered an art. Problem frames are discussed by [25] and it may be considered as abstract patterns of context diagrams for common classes of software problems and can also be reusable [1]. Considering the reuse of requirements models, [24] propose the reusability of UML artifacts through an artifact library. They present a proposal of reuse with focus on sequence diagrams. In addition, [23] propose models based on wikis to tackle reuse in software projects. Some approaches [18], [16], [9], [2] and [14] try to establish guidelines and metrics as a way of planning the management of the requirements for application of a systematic reuse plan. In [28] the authors propose ontology as a method of elicitation and analysis of requirements. The ontology can be used to obtain the knowledge of the domain and gives semantic to the requirements through semantics functions and inference rules. In [3] the author designed a tool to streamline the process of specifying a software system by automating processes. The author assures that the tool can help to reduce errors for the orga-

nization of requirements modeling and that includes reuse functionalities. As an answer to the QF established in this work, several approaches have been applied as a means of establishing methods and making systematic reuse of software requirements possible. The main approaches that are being used for repositories management as well as for documentation and standardization of requirements are: patterns; feature models; problem frames; natural-language;

wikis and ontologies. Although these approaches search for ways of establishing systematic reuse of requirements, there is still the need for requirements integration, project and code. This will make reuse feasible during the whole development process. For this, it is initially necessary to identify how to model these assets adequately for this goal in RE field.

Source	Methodologies, Strategies or Advices	Kind of Solution	Application
[20]	Metamodeling variability to enable requirements reuse in product line	Solution Design	Action Research
[1]	Research directions in requirements engineering	Problem Investigation	Literature Review
[2]	Adopting a Standard Process to requirements elicitation	Solution Implementation	Action Research
[3]	Tool for the organization of requirements modeling in requirements specification	Solution Implementation	Design and Creation
[5]	Agent-based distributed software systems in product line	Solution Implementation	Experiment
[6]	Criteria for Comparing Requirements Variability Modeling Notations for Product Lines	Solution Selection	Survey
[7]	Requirements derivation from the Product Line	Solution Design	Action Research
[8]	Management natural-language requirements specifications in a software product line context	Implementation Evaluation	Case Study
[16]	Metrics to evaluate requirements reuse in Analysis Phase of Domain Framework Development	Solution validation	Action Research
[9]	How to provide reusable requirements to introduce knowledge reuse in requirements engineering	Problem Investigation	Literature Review
[10]	Use patterns to requirements elicitation, specification and validation	Solution design	Case Study
[13]	Product-line-oriented approach to reusing requirements	Solution design	Case Study
[14]	Reuse, standardization, and transformation of requirements to storage of reusable elements	Solution Implementation	Design Creation
[18]	Reengineering to management system requirements	Solution Selection	Action Research
[23]	Using wikis to tackle reuse in Software Projects	Implementation Evaluation	Case Study
[24]	Reuse UML artifacts in requirements engineering	Solution Implementation	Survey
[25]	Architecture-based problem frames in product line	Solution design	Case study
[28]	Towards a Multiple Ontology Framework for Requirements Elicitation and Reuse	Solution selection	Design Creation

**Table 2: Studies classification.**

## 7. Conclusions and Future Works

The Systematic review process done in this work has contributed significantly for the identification of gaps in requirements reuse research. The current state of the art in Requirements Reuse shows that there is a large diversity of techniques and strategies for modeling of requirements. This diversity raises difficulties for the process of reuse in a systematic way. The simple fact of storing requirements in repositories does not guarantee their reuse. To make systematic reuse feasible, it is also necessary that the operational support directed to its execution be adequate. Moreover, an efficient way to represent and store specifications

should be established, allowing their comparison, adaptation, and management as reusable elements. We should always keep in mind the goal of reuse during the whole development process. Hence, the main problems related to Requirements Reuse which may be considered as research hotspots and future works are:

- The diversity of techniques and the lack of modeling pattern and storing of the specificities aimed at reuse;
- The requirements reuse is still carried out in a non-systematic way;
- The feature models do not represent the related requirements thoroughly;



- The Variability modeling techniques need to be more focused in strategies for reuse of requirements;
- An efficient process for comparing and adapting the reusable requirements is still lacking at the stage of software development with reuse;
- The means of integrating requirements with other artifacts of software development are needed, so as to make systematic reuse possible during the whole developmental process;

In conclusion, we can see that requirements engineering is still not ready to reuse. Thus, it is necessary to present feasible alternatives to establish a systematic way for reuse of requirements. The next step of this research is to choose one or more of these hotspots to be the focus of our work.

## References

- [1] B. H. C. Cheng and J. M. Atlee. Research directions in requirements engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 285–303, Washington, DC, USA, 2007. IEEE Computer Society.
- [2] M. Daneva. Erp requirements engineering practice: Lessons learned. *IEEE Softw.*, 21(2):26–33, 2004.
- [3] S. Dascalu, E. Fritzinger, N. Debnath, and O. Akinwale. Storm: Software tool for the organization of requirements modeling. *Electro/information Technology, 2006 IEEE International Conference on*, pages 250–255, May 2006.
- [4] J. C. de Almeida Biolchini, P. G. Mian, A. C. C. Natali, T. U. Conte, and G. H. Travassos. Scientific research ontology to support systematic review in software engineering. *Adv. Eng. Inform.*, 21(2):133–151, 2007.
- [5] J. Dehlinger and R. R. Lutz. A product-line requirements approach to safe reuse in multi-agent systems. *SIGSOFT Softw. Eng. Notes*, 30(4):1–7, 2005.
- [6] O. Djebbi and C. Salinesi. Criteria for comparing requirements variability modeling notations for product lines. *Comparative Evaluation in Requirements Engineering, 2006. CERÉ '06. Fourth International Workshop on*, pages 20–35, Sept. 2006.
- [7] O. Djebbi, C. Salinesi, and D. Diaz. Deriving product line requirements: the red-pl guidance approach. In *APSEC '07: Proceedings of the 14th Asia-Pacific Software Engineering Conference*, pages 494–501, Washington, DC, USA, 2007. IEEE Computer Society.
- [8] M. Eriksson, J. Brstler, and K. Borg. Managing requirements specifications for product lines - an approach and industry case study. *Journal of Systems and Software*, 82(3):435 – 447, 2009.
- [9] A. Gregoriades, J.-E. Shih, and A. Sutcliffe. Human-centred requirements engineering. *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 154–163, Sept. 2004.
- [10] L. Hagge and K. Lappe. Patterns for the re process. In *RE '04: Proceedings of the Requirements Engineering Conference, 12th IEEE International*, pages 90–99, Washington, DC, USA, 2004. IEEE Computer Society.
- [11] J. P. Higgins and S. Green, editors. *Cochrane Handbook for Systematic Reviews of Interventions Version 5.0.1 [updated September 2008]*. The Cochrane Collaboration, 2008.
- [12] B. Kitchenham. Procedures for performing systematic reviews. Technical report, Keele University and NICTA, 2004.
- [13] R. T. Kolagari and M.-O. Reiser. Reusing requirements: The need for extended variability models. In *FSEN*, pages 129–143, 2007.
- [14] M. A. Laguna, O. López, and Y. Crespo. Reuse, standardization, and transformation of requirements. In *ICSR*, pages 329–338, 2004.
- [15] O. López, M. A. Laguna, and F. J. G. Peñalvo. A metamodel for requirements reuse. In *JISBD*, pages 427–428, 2002.
- [16] S. N. Matos and C. T. Fernandes. Measuring reuse during the analysis phase of domain framework development. In *ICSEA '07: Proceedings of the International Conference on Software Engineering Advances*, page 7, Washington, DC, USA, 2007. IEEE Computer Society.
- [17] M. D. McIlroy. Software engineering: Report on a conference sponsored by the nato science committee. In *NATO Software Engineering Conference*, pages 138–155, 1968.
- [18] L. Melikhova, A. Elcock, A. Dovzhikov, G. Bulatov, and D. Vavilov. Reengineering for system requirements reuse: Methodology and use-case. *Consumer Electronics, 2007. ISCE 2007. IEEE International Symposium on*, pages 1–4, June 2007.
- [19] H. J. H. Mohamed Shehata, Armin Eberlein. Requirements reuse and feature interaction management. In *15th International Conference Software and Systems Engineering and their Applications*, Paris, 2002. ICSSEA.
- [20] B. Moros, C. Vicente-Chicote, and A. Toval. Remm-studio +: Modeling variability to enable requirements reuse. In *ER '08: Proceedings of the 27th International Conference on Conceptual Modeling*, pages 530–531, Berlin, Heidelberg, 2008. Springer-Verlag.
- [21] B. Nuseibeh and S. M. Easterbrook. Requirements engineering: a roadmap. In *ICSE - Future of SE Track*, pages 35–46, 2000.
- [22] B. J. Oates. *Researching Information Systems and Computing*. Sage Publications Ltd., 2006.
- [23] J. Rech, C. Bogner, and V. Haas. Using wikis to tackle reuse in software projects. *IEEE Software*, 24(6):99–104, 2007.
- [24] W. N. Robinson and H. G. Woo. Finding reusable uml sequence diagrams automatically. *IEEE Software*, 21(5):60–67, 2004.
- [25] C. Wang, Q. Depei, and L. Chuda. Architecture-based problem frames constructing for software reuse. In *IWAAPF '06: Proceedings of the 2006 international workshop on Advances and applications of problem frames*, pages 19–24, New York, NY, USA, 2006. ACM.
- [26] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requir. Eng.*, 11(1):102–107, 2005.
- [27] P. Zave. Classification of research efforts in requirements engineering. *ACM Comput. Surv.*, 29(4):315–321, 1997.
- [28] L. Zong-yong, W. Zhi-xue, Y. Ying-ying, W. Yue, and L. Ying. Towards a multiple ontology framework for requirements elicitation and reuse. In *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference*, pages 189–195, Washington, DC, USA, 2007. IEEE Computer Society.

# Reprioritizing the Requirements in Agile Software Development: towards a Conceptual Model from Clients' Perspective

Zornitza Racheva, Maya Daneva

University of Twente, Netherlands, {z.racheva, m.daneva}@utwente.nl

**Abstract.** *Continuous and client-centric requirements reprioritization forms the very core of today's agile approaches. In this paper, we report on results of a grounded theory study on agile requirements prioritization methods. The outcome is a conceptual model for understanding the inter-iteration prioritization process from client's perspective. The latter is derived from the authors' experiences and by using empirical data, published earlier by other authors.*

**Keywords:** agile development, requirements prioritization, inter-iteration decision-making process, grounded theory.

## Introduction

Continuous requirements reprioritization from client's perspective forms the key of today's agile approaches. A recent empirical study [7] indicates that, with respect to requirements (re)prioritization, agile RE differs from 'traditional RE' in two ways: (i) (re)prioritization happens at inter-iteration time, which means the project team anticipates and plans as many reprioritization sessions as the number of project iterations, and (ii) (re)prioritization is based mostly on business value, that is, the highest priority *features* (i.e. requirements in agile terminology) get implemented early so that most business value gets realized. These two aspects of agile RE pose at least two challenges: (i) continuous reprioritization more often than not (especially when practiced without caution) leads to project instability, and (ii) clients, by and large, relate the concept of business value to features that meet their functional requirements, so non-functional requirements (such as scalability or security) that might initially appear secondary to clients, turn out critical for the operational success of the product. For example, redesigning the architecture of the software product at a late stage would add up to an over-expensive or a delayed project. In a context of a supplier network, these challenges may well aggravate further, as product managers make commitments based on process and product assumptions which are different from the ones of the development team. Yet, these assumptions might be essential to product success.

Our paper is a first attempt to respond to these two challenges. It proposes a conceptual model of the agile prioritization process from client's perspective. We

obtained it by applying a grounded theory approach. We make the note that we do not provide a new prioritization technique. Instead, we (i) redefine our view of requirements and their (re)prioritization by treating them from a clients' perspective, and (ii) we propose a model that reflects this specific focus and represents an unified approach to discussing the prioritization effort independently from the particular method that is used.

Our motivation for creating this model originates on the premise that the practices of continuous requirements reprioritization, with strong client participation, are a relatively recent phenomenon and because of this are only partially understood. As agile literature indicates, never before in the software engineering history, the client has been that actively involved in the requirements reprioritization as he/she is in agile. When the client is expected to actively participate in the process by performing, among other task, the key task of prioritizing requirements, he or she must be aware of the facets of his/her role and thus would profit from a decision-support vehicle available at his/her disposal. We think that a conceptual model can help the client in multiple ways: (i) to navigate through the agile process of delivering business value; (ii) to make explicit the tacit assumptions, used in different requirements prioritization methods; (iii) to identify the possible peaces and sources of information that might be of importance for the outcome of the prioritization and, consequently, for the project; (iv) to make the process more objective in the sense that having such a vehicle will allow also less experienced users to participate in the prioritization process and, they could do it with the confidence that they deliver a quality work.

The paper is structured as follows: Section 2 provides a discussion on the role of clients in agile RE and formulates our research question. Section 3 presents the Grounded Theory research method and Section 4 – its application and results. Section 5 evaluates our results and discusses the possible threats to their validity. Section 6 concludes the paper.

## Client-driven Requirements Prioritization and Reprioritization

The agile manifesto [26] deems the client's role critical in making decisions about "what to build". In the minimalist philosophy of XP – a prominent agile approach, the following is recommended for the client's role [5]: (1) The client is an integral part of the team and

should be on-site with the team. (2) The client writes user stories and then discusses each requirement directly with the programmers. (3) The client is responsible for all business decisions including prioritizing user story development. (4) The small 2-3 week iterations allow the user to evolve their requirements based on concrete working software. (5) The client regularly tests the software to confirm it works as expected.

Our focus in this paper is on item 3 of the above list, namely supporting the client when making prioritization decisions. Therefore, in this and the next sections, we re-focus the discussion on the role of clients' requirements prioritization (RP) in agile software development.

Clearly, RP is a part of any project, independently from the developing method. Yet, the purpose and the place of this activity are essentially different when we distinguish between 'traditional' and agile development. In a 'traditional' (e.g. gated or waterfall-style life cycle), it is about which requirements (i) to implement earlier than others, or (ii) to include in an earlier release. The premise is that the whole functionality can not be implemented in the same time, but it will eventually be implemented. So it is a project-management activity from the developers' side. When asked about priorities in a 'traditional' project, the client tends to qualify the majority of the requirements as high priority.

In contrast to 'traditional' development, agile projects rest on the understanding, that the whole functionality will not be implemented and delivered at once with the first release, and part of it will be eventually not implemented. The problem, then, is: (i) how to decide on what to implement in each (next) iteration, and (ii) which requirements will deliver the maximum value to the clients as early as possible. One of the biggest assets of an agile approach is that business value is delivered to the client throughout the project, and the return on investment is generated much earlier. Thus any changes in the requirements can be taken into consideration and implemented into the product at an early stage. This highlights the paramount importance of the RP activities.

The changes in the list with requirements for an iteration might occur for different reasons – new market or company realities or better knowledge about the value certain features deliver. This requires a dynamic prioritization process as well. This view is supported by Harris and Cohn [19], who use tactics to minimize costs and maximize benefits through strategic learning and provide guidelines on how to optimize business value. They prove the necessity of adopting a dynamic approach to agile prioritization, in order to take into consideration the important aspect of learning in an agile project. Their focus is particularly on incorporating learning and cost of change in the decision-making process.

Last, while in a traditional project the prioritization is usually performed once and before the implementation phase, in agile context it is an ongoing process, performed

in the beginning of each iteration, or even during the iteration; this reflects the dynamics of the project's backlog. The differences between the two settings are summarized in Table 1.

Table 1. Comparison of traditional and agile RP

Aspects of the RP process	Traditional (waterfall) development	Agile
Goals/Purpose of the RP	Project management-vehicle	- Vehicle to ensure that delivered business value is maximized at each iteration; - Scope definition vehicle at iteration level
When is RP performed	Typically once, after the analysis phase and before implementation	Before each iteration, at planning phase, or during iteration
Who is responsible for RP	Developer, with participation of project manager and other stakeholders.	Client is the key driver for choosing, being supported by Scrum Master (or Agile Project Manager) regarding the assessment of the technical feasibility of a schedule.

Building on the above discussion, we came up with the following research question (RQ): *“What are the key topics to consider when prioritizing the requirements from client’s perspective in agile projects?”*

## The Grounded Theory Approach

The term grounded theory [8] refers to a set of systematic guidelines for data gathering, coding, synthesizing, categorizing, and integrating concepts to conduct a theoretical analysis of an empirical problem. The name grounded theory points out to its fundamental premise that a researcher can and should develop theory from rigorous analyses of empirical data. As a qualitative research method, GT is distinctive in (i) that it is inductive in nature, which means that we as researchers have no preconceived ideas to prove or disprove data, (ii) that collection and analysis proceed simultaneously and each informs the other, and (iii) that constant comparative techniques treat possible disagreements between the emerging theory and newly collected information. A GT exercise of a studied topic starts with concrete data and ends with rendering them in an explanatory theory. From the very beginning, the researcher analyzes the data and identifies analytic leads and tentative categories to develop through further data collection. It is essential to note that, in this process, whenever the emerging theory disagrees with newly collected information from experiences or from literature, the researcher should not assume that the theory is wrong. Instead, the researcher seeks to extend the theory so that it makes sense of both

the data from the study and the data from the literature, because the key concern throughout a GT process is the fit of the theory to the data and its ability to make sense of actual experience.

Research methodologists [8,17,31] suggest that a theory derived from data is more likely to resemble what’s happening in reality, than a theory which is derived by putting together a set of concepts based on experience and solely through assumptions about how things in real life would work. As GT studies rest on the data, they are thought to enhance researchers’ understanding of a situation and provide a meaningful starting point for further action. The philosophical foundation of GT and how it affects the researcher’s choices in carrying out his/her work have been discussed in [8] and are beyond the scope of this paper. Here, we focus on the application of the GT process [8] and the results we obtained.

## The Application of Grounded Theory

For the purpose of our research, we used the GT guidelines by Kathy Charmaz [8]. We executed a research process which included the following steps: (1) identification and review of data sources from published literature, (2) initial and focused coding, (3) clustering and memo-writing, (4) conceptual modelling, and (5) theoretical sampling of empirical data, using the concepts and categories from our resulting conceptual model. The goal of steps 1-3 is the discovery of as many relevant categories as possible, along with their properties and dimensions. Step 4 is about the visual representation of the categories and their relationships, and Step 5 is about ‘saturating the categories’. Categories are considered ‘saturated’ when collecting fresh data no longer brings new theoretical insights nor reveals new properties of the categories in the conceptual model [8].

We traversed the steps 1-5 multiple times, as methodologists recommend [8], because: “Constant interplay between proposing and checking [...] is what makes our theory grounded!” [31]. That means, the analysis of the data collected in one step helps to check the interpretations from the previous step. In the subsections below, we indicate the execution of the steps along with the results we obtained from our application of GT.

### 4.1. The sources

In this study, the data used and constantly compared to the emerging theory is literature on agile requirements prioritization available via scientific digital libraries and prominent agile practitioners’ journals. We did a semi-systematic literature search using the five bibliographic databases: IEEEExplore, ACM Digital Library, Google Scholar, InterScience and Citeseer. We complemented

them with the following periodicals: the Agile Journal [1], and the platforms, dedicated to software development and agile methods: DrDobb’s [13] and InfoQ [20]. The key words we used for our search were: *agile, requirements, backlog, prioritization, inter-iteration, decision-making, business value, risk, cost, features*. We traced the references in the identified papers to get access to other relevant sources. To determine whether to include or not these sources to our GT research, for each one, we reviewed the abstracts and the conclusions and we checked this information against the following five quality criteria for inclusion in the review: (1) the paper is on a agile RP, (2) the paper is credible, i.e. the method described is meaningful and intuitive to follow; (3) relevance for practice: the RP method potentially offers support for practical requirements prioritization, (4) the paper adequately describes the context, in which the method is expected to be applicable; ‘adequately’ means that the reader can replicate the use of the RPM in his/her own context; and (5) original paper: for each method, we searched at least its original publication; if an original paper is difficult to access, or is outside the RE field, we included another description from an RE author. The publications were written in English only and included both qualitative and quantitative research, from scientists and practitioners. We carried out the quality check by using these criteria, which yielded 42 papers eligible for inclusion and review in the GT process. These papers refer to 15 RP methods and one technique, as indicated in Table 2.

**Table 2. The RP approaches published in the sources used for the GT**

RP method	References
Round-the-group prioritization	[6]
Ping Pong Balls	[30]
\$100 allocation (cumulative voting)	[22]
Multi-voting system	[32]
MoSCoW	[16]
Pair-wise analysis	[18] [21]
Weighted criteria analysis	[18]
Analytic Hierarchy Process (AHP).	[29]
Dot voting	[18]
Binary Search Tree (BST)	[2]
Ranking based on product definition	[15]
Planning Game	[5] [21]
Quality functional deployment QFD	[11][18]
Wiegiers’ matrix approach	[33]
Mathematical programming techniques for release planning	[23]
Technique of bucketing requirements	[25]

## 4.2. The Conceptual model

The multiple iterations of coding, constant comparing of information from literature, and conceptual modelling in our GT process delivered two models, Model A, which is presented in Fig 1 and Model B, which is presented in Fig 2. Model A describes the agile RP process, while Model B elaborates on the conceptual categories related to making the RP decisions. We make the note that Model B (on Fig 2) is not meant as a refinement of Model A (on Fig 1). Instead, the purpose of Model B is to explicate and bring insights into the decision-making step, which is the core of the RP process.

Furthermore, both models take the perspective of the client, unlike RP authors [4] who adopt the perspective of the development team. We must note that the models take a 'big-picture' view to make explicit those pieces of information, necessary for the prioritization process.

To create these models we used the initial and focused coding practices described in [8]. This meant first to name the segments of data, and then to use the most frequent initial codes to "sort, synthesize, integrate and organize large amounts of data" [8]. We make the note that to us, the focused coding meant iteratively making decisions about those initial codes which the two authors deemed to make the most analytic sense to categorize the data, as Kathy Charmaz says, "incisively and completely". We complemented our coding with diagramming, which enabled us to visualize the connections among the conceptual categories and to see more clearly the relative strength or weakness of the relationships between the concepts. Our intensive diagramming activity was motivated by Adele Clarke [9] who contends that conceptual mapping preserves empirical realities and complexities. We drew diagrams and wrote notes, then reviewed them and dissected them meaningfully, while keeping the relations between the parts (that are dominant concepts, themes, and issues) intact. We followed this process, as it is meant to help the researcher to reduce and analyze data and direct him/her toward trends, themes, and patterns. Due to space

limitation, we do not provide a mapping between the literature sources we used to as input to build the model and the parts of the model derived from each source. We, however, plan to publish this in a separate paper in near future. Below, we describe the two models in more detail.

### 4.2.1. Model A

This model presents a generic prioritization process in terms of its inputs and outputs, as it is visible from the client's standpoint. We deliberately used concepts which make clear how the status of requirements changes – from 'Initial' to 'Prioritized', to "Sprint", to 'Implemented'. The input is the *Initial Project Backlog*, that is the total number of requirements upon the start of the project. Before the very first agile iteration, the client runs a RP technique, which produces *Prioritized Project Backlog*. This is an ordered list of the requirements (originally written in the *Initial Project Backlog*) according to their priorities. In agile settings, only a small portion of the upper part of this ordered list goes for implementation in the first iteration. (Iterations are called *sprint* in the jargon of Scrum - the most popular agile project management approach.) This small portion of prioritized requirements forms the so-called *Sprint Backlog*. Once the iteration is completed, the status of those requirements which are already implemented in the software product, changes to *Implemented requirements*. Those requirements which could not be implemented by the developer team are fed back into the project backlog and are subject to reprioritization before the new iteration starts. At that inter-iteration time, the client might decide to request a change to the requirements and this leads to a new reprioritization as well (this is the arrow from *Sprint backlog* to *Prioritized project backlog*). The client reprioritizes the project backlog, so that she/he knows the next portion of requirements which will go to the next *Sprint Backlog*. The relationship between the concepts *Prioritized Project Backlog* and *Sprint Backlog* - from the view point of the clients in agile projects, is elucidated in Model B (see Fig 2).

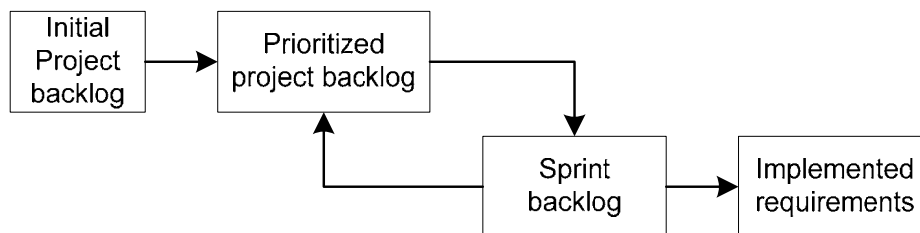


Fig. 1. Model A: the prioritization process from clients' perspective.

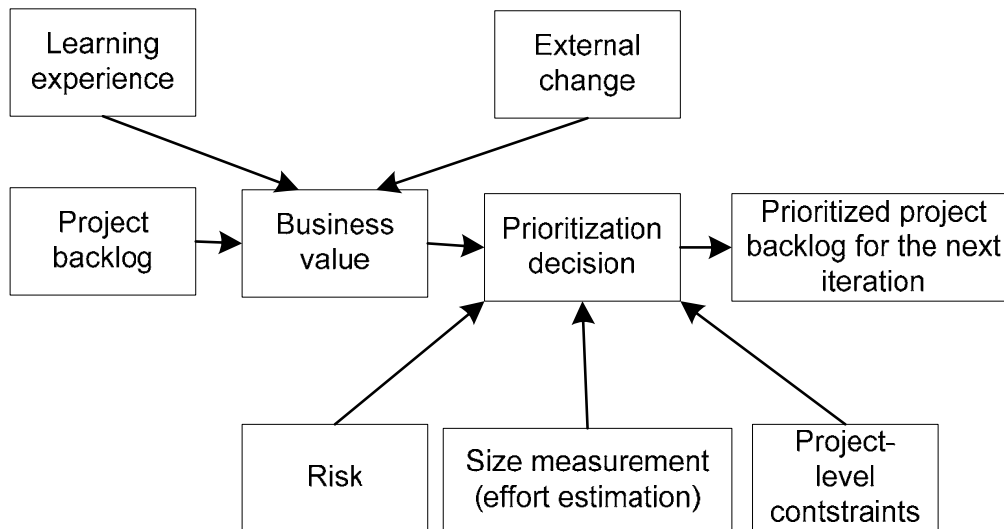


Fig. 2. Model B: topics to consider when making prioritization decisions

#### 4.2.2. Model B

This model is to help clients ‘zoom-in’ and see the aspects important for RP at inter-iteration time. As in Model A, in Model B we take a holistic perspective of RP. In contrast to Model A, Model B can be seen as a generic *framework for describing* the client’s decision-making situation while prioritizing the requirements. As per Alenjung and Person [3], a decision-making situation is “a contextual whole of related aspects that concerns a decision-maker”, that is – in our case, the client in an agile project. For example, one can use the conceptual categories of the framework (that is, Model B) to depict a specific client’s RP situation in a specific agile project, in a specific organization and, thus, take into account the topics important to be considered by the client when prioritizing requirements at inter-iteration time.

Furthermore, Model B shows the complexity of the decision-making from client’s perspective in agile RP. We observe, that the client, when prioritizing the *Project backlog*, explicitly or implicitly relies on tacit knowledge to estimate the *Business value* of each item (in the *Project backlog*). The estimation is qualitative (as it was already found in our previously published study on business value in agile [27]). Yet, the agile clients make a conscious effort to connect business value to “something that delivers profit to the organization paying for the software in the form of an increase in revenue, an avoidance of costs, or an improvement in service” [24].

The client assesses the *Business Value* of the requirements in the project backlog based on his/her current knowledge and *Learning Experiences* within the agile project as well as any changes (see the box *External*

*changes* in Fig 2) occurring in the business environment of the organization. An example for an external change can be a merger between the client’s organization and another organization. Both the client’s continuous learning throughout the project and the dynamic environment in which the client organization operates can make - from one iteration to another, some requirements more valuable than others. (It is possible that *External changes* can even render some requirements irrelevant).

Model B suggests that there are four aspects which the client considers when making his/her decision on requirements priorities: *Business Value*, *Risk*, *Size Measurement/Effort Estimation*, and *Project-level Constraints*. These four aspects are important to the way and the possibilities for a client to execute the decision-making step. We make the note that the agile RP literature sources converge on that the *Business value* is the dominating RP criterion. We also observed that some RP methods used the notion of ‘importance’, or relative importance of a feature, compared to other requirements, instead of ‘value’. Still, when reading about the application of the RP method, we understand that estimations of ‘value’, is the implicit prerequisite for these prioritization methods. In addition to Business Value, the client considers *Risk* due to development instability. Accommodating highly-volatile requirements, which in turn, means accommodating instabilities in the development process is an inherent aspect of the agile development process. As a matter of fact, the strong focus on business value and on continuous reprioritization of the requirements is the key to successfully coping with instability and volatile requirements.

Next, the client considers *Estimated Effort* based on functional size when making decisions on priorities for the next iteration. Size is based on the user stories and can, for example, be expressed in story points [10].

Another aspect which can be a consideration during the decision-making is a *Project-level Constraint*. This can include, e.g. budget constraints, fixed market-driven deadline or human resource constraint.

Last, the *Prioritized Project Backlog* is the ordered list of requirements which the developer team should act upon in the next iteration.

We make the note that Model B fits the contextual whole of those related aspects which concern the client when using any of the 15 RP techniques covered in the literature sources for our GT study (see Table 2). This means that a client could use Model B to reason about his requirements prioritization context when using any of these techniques. Clearly, not all of the elements in Model B are necessarily present in each prioritization effort – i.e. some of them are optional depending on the project’s context or on the method used. For example, *Risk* (due to instability and highly volatile requirements) is usually a serious consideration in the later project iterations, for example when a large portion of the budget has already been consumed, or when a critical delivery deadline is approaching.

### 4.3. Theoretical sampling and saturating the concepts

This section briefly discusses the purpose of our theoretical sampling and how we carried it out. As per methodologists [8, 12, 17, 31], theoretical sampling means a quick and focused collection of pinpointed data once the researcher has a first set of conceptual categories to direct his/her theoretical sampling. In our study, we tentatively conceptualized relevant ideas which hinted to areas to probe with more information. We selectively looked for people and online forums on agile software development to shed light into what could be the boundaries of our conceptual categories. To get access to people, we used our own professional networks and agile-focused workshop venues (for example, the agile workshop co-located with the International Conference on Software Engineering in 2008 in Leipzig, where the authors presented the very first draft of the conceptual model in Fig 2). Specifically, we involved three practitioners from companies, when we were trying to figure out (i) how clients define, estimate and use size in agile RP context, (ii) how clients define business value, and (iii) how clients (or product owners) manage sprint backlogs in the context of agile projects in supplier networks. These practitioners (Luigi Buglione from large IT-solution providing company, Thijs Munsterman from a mid-sized agile software development company, and Erlend Engum from a small agile developing company, who helped out in understanding (i), (ii) and (iii)

respectively) brought insights into the variation in the meanings of our conceptual categories (*Size Measurement*, *Business Value*, and *Risk*). Checking our concepts against the empirical realities of the practitioners was instrumental to understand how, when, and why the meanings of our categories vary.

Similarly to this, our screening of published experiences in prominent agile blogs (for example [28]) and forums (for example [14]) contributed to the identification of those categories which we overlooked (for example *Project-level Constraints* and *Learning Experience*), or under-analysed (for example *Risk*).

We stopped our theoretical sampling process when we noticed that further acquisition of data from real project experiences did not bring new ideas nor opened up new ways to think of the properties of our conceptual categories. In GT, this state is called ‘saturation’ of the resulting conceptual model [8]. We however, acknowledge that this judgement about the point at which we stop the theoretical sampling might be subjective. Therefore, in the immediate future, we are planning case studies on real projects with companies in which we will use Model B as our framework to describe the contextual whole of the related aspects that concern the client when prioritizing the requirements at inter-iteration time.

## Evaluation of the GT results

Research methodologists [12, 17, 31] emphasize that when a researcher builds up a theory by using a qualitative approach as the GT, it makes more sense for the researcher to assess its resulting theory in terms of explanatory power than in terms of generalizability. As a conceptual model based on GT is always context-dependent (and this is reflected in the categories), methodologists do not propose that the GT findings are generalizable beyond the defined boundary of the study. To study explanatory power, we considered Glaser’s three key criteria for evaluating the emerging theory: adequacy, fitness (or relevance) and modifiability. Adequacy is to be assured by applying the set of techniques and analytical procedures in the GT, for example, adhering as closely as possible to the GT principles and processes, coding the data independently by each researcher before re-coding them in joint work discussions (in order to ensure the highest possible degree of inter-coder reliability), consulting literature to evaluate similarities and dissimilarities of the resulting theory to extend literature and to check for any category, property or property value that might have been overlooked. We made conscious effort to keep these GT principles, however, we must be clear on a validity concern arising from the fact that most of the time the two authors worked away from each other at two different locations and could not do much joint re-coding.

The relevance of the results to researchers is to be judged regarding how it fits the situation, that is, whether it helps individuals familiar with the phenomenon (in this study, requirement prioritization) - either as researchers or as 'lay observers' - to make sense of their experience and to manage the situation better. To make sure we preserve the meaning of the clients in agile projects, we made the conscious choice to search and include the so-called 'in-vivo' codes, as recommended by Kathy Charmaz [8]. These are special terms from the world of the individuals involved in the studied context, which are assumed that everyone "knows and shares" them, which flag condensed but essential meaning, and which reflect assumptions that frame some actions. In our case, examples of in-vivo codes, associated to clients in agile software development, were "backlog" (meaning those requirements in an agile project, which are subjected to the implementation - for the whole project, as well as for immediate future iteration, 'project backlog' and 'sprint backlog' respectively) and "sprint" (meaning an individual agile iteration in a project). We looked into the implicit meaning behind these terms and this in fact was what brought us to Model A on Fig 1. Another measure we took in order to keep our conceptual modelling effort in sync with real experiences was our consistent engagement in diagramming activity, details on which were presented in section 4.2.1. Beyond these two steps (using in-vivo codes and diagramming), in our immediate future research, we plan to demonstrate the fit of the framework by using it in case studies.

Furthermore, modifiability of an emerging theory is concerned with the possibility to update it and extend it in the future. We made a conscious effort to maintain a balance between keeping the concepts abstract enough - so that the theory can serve as a general explanation, and making sure the concepts do not get too abstract as to lose their sensitizing characteristics. In our view, we should keep our framework open as it makes more sense to invite other researchers to use it and test it, only after this, to strive for all-inclusive and general results. We do think that if industrial uptake of agile software development practices increases and more knowledge on the client's role and the client-develop interaction modes becomes available, our framework will need some refinement and extension so that it's kept useful.

Last, we point out like other qualitative research approached, the GT approach implies the risk that the researchers assume that the conceptual categories are saturated, when they might not be. Following Charmaz [8], we remained open at all times to any new literature source and whenever we felt we were getting stuck, we stepped back and re-coded the earlier collected information and looked for new leads. We also looked at many cases of agile RP, while carrying out the theoretical sampling and this increased our understanding of the empirical world and helped us discern variations in the

conceptual categories we use to describe the agile RP from client's perspective.

## Conclusions

The contribution of this work is a conceptual framework which is a grounded theory explicating the requirements reprioritization in agile software development. This conceptual model fills a gap in the current agile software engineering and agile requirements engineering literature which lacks comprehensive studies on agile prioritization. Our conceptual model is a first proposal only. However, we think that it opens up for other researchers to explore the area and to accumulate support for - or a challenge to, the proposed theory. Our immediate future step is to carry out case study research in agile companies in the Netherlands.

## Acknowledgement

This research has been funded by the Netherlands Research Foundation (NWO) under the QUADREAD project and under CARES project. The authors would like to thank the practitioners Luigi Buglione, Thijs Munsterman, Erlend Engum, the colleagues Roel Wieringa, Klaas Sikkel, Klaas van den Berg, Siv Hilde Houmb, and the participants of the APSO workshop at ICSE 2008 for sharing ideas on the topic of agile requirements prioritization. We also thank the anonymous SEKE reviewers whose comments and suggestions brought us to an improved version of this paper.

## References

- [1] Agile Journal <http://www.agilejournal.com/>
- [2] Ahl, V. "An Experimental Comparison of Five Prioritization Methods." Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden, 2005.
- [3] Alenljung, B, A. Person, Portraying the practice of decision-making in requirements Engineering: a Case Study of large Scale Bespoke Development, Requirements Engineering journal, 2008, 13, pp. 257-279.
- [4] Augustine, S., Managing Agile Projects, Prentice-Hall, 2005.
- [5] Beck, K. eXtreme Programming Explained: Embrace Change, Addison Wesley, 2000.
- [6] Berteig, M., Methods of Prioritization, March 20, 2006 in Agile Advice Online Practitioners Forum, [http://www.agileadvice.com/archives/2006/03/methods\\_of\\_prio.html](http://www.agileadvice.com/archives/2006/03/methods_of_prio.html)
- [7] Cao, L, Ramesh B., Agile Requirements Engineering Practices: An Empirical Study, IEEE Software, Jan/Feb, 2008 pp. 60-67.
- [8] Charmaz, K. Constructing Grounded Theory: a Practical Guide through Qualitative Research, Thousand Oaks CA, Sage, 2007.
- [9] Clarke, A. Situational Analysis: Grounded Theory after the Postmodern Turn, Thousand Oaks, CA, Sage, 2005.



- [10] Cohn, M "Agile estimating and planning", Prentice Hall,20
- [11] Crow, K.,: "Customer-focused Development with QFD", URL: <http://www.npd-solutions.com/qfd.html>
- [12] Dey, I. Grounding Grounded Theory, San Diego Academic Press, 1999.
- [13] Dr Dobb's portal <http://www.ddj.com/>
- [14] Forum, Agile Journal, <http://www.agilejournal.com/forums>
- [15] Fraser, J., Setting Priorities, April 23, 2002, URL: <http://www.adaptivepath.com/ideas/essays/archives/000018.php>
- [16] Getting Started With Use Case Modeling, An Oracle White Paper, May 2007 <http://www.oracle.com/technology/products/jdev/collateral/papers/10g/gswUseCaseModeling.pdf>
- [17] Glaser B. G., Basics of grounded theory analysis:emergence vs forcing, Mill Valley, Ca.: Sociology Press,1992.
- [18] Gottesdiener, E., At a Glance: Other Prioritization Methods, EBG Consulting, Inc. [www.ebgconsulting.com](http://www.ebgconsulting.com)
- [19] Harris, R. S., M. Cohn: Incorporating Learning and Expected Cost of Change in Prioritizing Features on Agile Projects. XP 2006: pp. 175-180
- [20] InfoQ software development community <http://www.infoq.com/agile>
- [21] Karlsson, L., Thelin, T., Regnell. B., Berander, P., Wohlin, C., Pair-wise comparisons versus planning game partitioning--experiments on requirements prioritisation techniques, Empirical Software Engineering, 12(11), 2007
- [22] Leffingwell, D., Widrig, D., Managing Software Requirements, 2nd ed. Boston, MA: Addison-Wesley, 2003
- [23] Li, C., Akker, J.M. van den, Brinkkemper, S. & Diepen, G. (2007). Intergrated Requirement Selection and Scheduling for the Release Planning of a Software Product. In Proc. of REFSQ '07, Spingre LNCS, pp. 93-108.
- [24] Patton ,Jeff. Ambiguous Business Value Harms Software Products, IEEE Software, 25(1) , January/February 2008.
- [25] Patton, J., Finding the forest in the trees, Conference on Object Oriented Programming Systems Languages and Applications, 2005, San Diego, CA, USA, pp: 266 – 274.
- [26] Principles behind the Agile Manifesto, 2001, URL: <http://agilemanifesto.org/principles.html>
- [27] Racheva, Z., M. Daneva, K. Sikkil, Value Creation by Agile Projects: Methodology or Mystery? to appear in the Proceedings of PROFES 2009 Conference, June 15-17, Oulu, Finland, Lecture Notes of Computer Science
- [28] Rally Dev Agile Blog <http://www.rallydev.com/agileblog>
- [29] Saaty, T.L., The Analytic Hierarchy Process, McGraw-Hill, New York, 1980.
- [30] Schwaber K., Agile Project Management with SCRUM, Microsoft Press, 2004.
- [31] Strauss, A.L., J.M. Corbin, Basics of qualitative research - grounded theory procedures and techniques, 6th print, Sage, Newbury Park, USA, 1991.
- [32] Tabaka, J., Collaboration Explained: Facilitation Skills for Software Project Leaders. Addison Wesley 2006.
- [33] Wiegers, K., "First Things First: Prioritizing Requirements," *Software Development*, 7(9) 1999.

# A Novel Hybrid Search Algorithm for Feature Selection

Pengpeng Lin<sup>1</sup>, Huanjing Wang<sup>1</sup> and Taghi M. Khoshgoftaar<sup>2</sup>

<sup>1</sup>Department of Mathematics and Computer Science, Western Kentucky University, USA

<sup>2</sup>Department of Computer Science and Engineering, Florida Atlantic University, USA

## Abstract

Data mining is the exploration and analysis of large datasets for discovering hidden knowledge and patterns. The various techniques from the field of data mining have been successfully applied to a variety of domains. An important area of data mining and machine learning is feature selection. The goal of feature selection is to find a minimum set of features (attributes) such that the reduced dataset characterizes the data similarly as the original dataset without significantly reducing the accuracy of the classifier. We propose a new feature selection algorithm called Automatic Hybrid Search (AHS) that generates consistent feature subsets and is a hybrid of the filter and the wrapper models. Our experiments have shown that AHS performed well at feature selection with a relatively lower runtime cost, a smaller size of the selected feature subset, and a lower error rate than the more traditional approaches such as exhaustive search, heuristic search, and probabilistic search. The findings suggest that AHS is more sensitive to the number of features than to the number of instances in the dataset.

## 1. INTRODUCTION

Data as the target for data mining has increased dimensionally in number of instances (i.e., size) and the number of features in a dataset. Data mining is the exploration and analysis of large datasets for discovering hidden knowledge and patterns. Various techniques from the field of data mining and machine learning have been successfully applied for deriving new information in a variety of domains [9]. The primary process of data mining is three-fold: data preprocessing, learning, and post-processing [2]. Among these, the first step is an essential preparation for the latter two. In this paper, we focus our attention on the data preprocessing phase.

Data preprocessing is an important step in the data mining process because of the need for high-quality data. Data quality is a multi-faceted issue for data mining, because poor data quality is often a problem in practical applications of data mining, and it affects the success of the data mining objectives, such as prediction, classification, clustering, association rules, description, and estimation.

Data preprocessing includes data cleansing, data integration, data transformation, and data reduction. In practice, it has been found that data preprocessing takes approximately 80% of the total data mining effort [11]. Real world data may be incomplete, noisy and inconsistent. Data cleansing works toward identifying inaccuracies and noise in data, and attempts to correct them. Data integration is the process of combining data residing at different sources and providing the user with a unified view of these data [10]. The goal of data transformation is to transform data into forms that are appropriate for mining. Data reduction obtains a reduced representation of the dataset that is relatively smaller than the original dataset, such that similar results can be obtained with reduced dataset as with the original dataset. Data reduction includes:

- Feature selection which involves keeping only useful features and removing irrelevant and noisy information, at the same time improving efficiency without significantly reducing accuracy of the classifier,
- Reducing the number of attributes' values by grouping them into intervals or grouping values in clusters, and
- Reducing the number of records (instances) in the dataset.

This paper focuses on feature selection, also called attribute selection, variable selection, or variable and feature selection [2]. A dataset for data mining may contain a large number of features, many of which may be irrelevant or noisy to the learning task. The goal of feature selection is to find a minimum set of features such that the reduced dataset describes the data as close as possible to original dataset without significantly reducing the accuracy of the subsequent classifier.

In order to select a subset of relevant features, an evaluation criterion must be implemented. The evaluation criterion is used to measure the goodness of the selected features. There are many searching strategies have been designed with various evaluation criteria. Feature selection algorithms designed with different evaluation criterion broadly fall into three categories: the filter model, the wrapper model, and the hybrid model [5]. The filter model

evaluates the feature subsets based on the general characteristics of data without involving any algorithm. The wrapper model requires an evaluation criterion with a predetermined learning algorithm. The hybrid model is a combination of filter model and wrapper model, and thus exploits advantage from both models.

We review an evaluation criterion called consistency measure. Using this measure, feature selection is formalized as: finding the smallest set of features that can distinguish classes as compared with the full (non-reduced) set [1]. Three existing search algorithms are examined and implemented. One new algorithm, named Automatic Hybrid Search (AHS), is proposed based on them. A consistency-based feature selection framework<sup>1</sup> is developed using Java for comparison purposes.

The remainder of the paper is organized as follows. We review the relevant literature on the consistency measure and existing search algorithms in Section 2. Section 3 then describes the proposed search algorithm. We discuss our experimental results in Section 4. Finally, at the end we conclude our paper in Section 5, and provide suggestions for future work.

## 2. RELATED LITERATURE

The problem of feature selection is to find a minimum subset of features according to the given evaluation criterion. By evaluating each selected subset, we can reduce the number of possible combinations; thus, simplify the classifier. Many existing evaluation criteria are accurate only for discrete data. The preprocessing step is needed to discretize data before applying these evaluation criteria. We use datasets that have already been discretized.

Dash and Liu [12] suggest a feature selection process consisting of four parts: feature generation, feature evaluation, stopping criterion, and testing. Feature generation uses a certain searching strategy to produce the candidate feature subset. Each selected subset is then evaluated by a criterion for its merit and compared with the previous best result. If the new selected subset has better merit than the previous best result, then the previous best subset is replaced with the new subset. This process of feature generation and evaluation is repeated until a stopping criterion is met. Finally, the testing procedure tests the selected feature subset. In general, a search algorithm and an evaluation function are needed for the feature selection process.

In the remainder of this section, Section 2.1 provides a review of an evaluation criterion called consistency measure, while Section 2.2 summarizes a review of existing search algorithms that use the consistency

measure.

### 2.1 Evaluation Criterion

An evaluation criterion is used to select the most relevant features, and thus, eliminating irrelevancy and redundancy. As described by Kohavi and John [4], feature relevance can be classified into three categories: strongly relevant, weakly relevant, and irrelevant. If a feature is strongly relevant, it indicates that the feature belongs to the optimal feature subset, and removing it will affect class distribution. Weakly relevant features are not always needed to obtain the optimal feature subset. Irrelevant features should not be considered and should be removed. There are a variety of evaluation criteria that can be used for controlling feature selection. We focus on a popular evaluation criterion called consistency measure [1].

The consistency measure is used to find a minimum feature subset that can consistently yield a classifier as if using the full feature set. Consistency rate [1] is defined by the inconsistency rate where two instances are considered inconsistent if having the same feature values but different class labels. To compute inconsistency rate [1], the inconsistency count is first computed. Assume that the target feature has  $j$  different class labels:  $C_1, C_2, \dots, C_j$ . For a feature subset  $S$  with  $M$  number of instances, there are  $h$  patterns,  $P_1, P_2, \dots, P_h$ . A pattern  $P_i$  ( $1 \leq i \leq h$ ) appears in  $N$  instances out of which  $N_1$  number of instances are labeled  $C_1$ ,  $N_2$  number of instances are labeled  $C_2$ , and so on. If  $N_1$  is the largest among the  $j$  classes, the inconsistency count  $INC_i = N - N_1$  for pattern  $P_i$ . In total, there are  $h$  inconsistency counts and the inconsistency rate is the sum of all the inconsistency counts over all patterns divided by total number of instances. The inconsistency rate  $INCR$  can be expressed as follows:

$$INCR = \sum_{i=1}^h INC_i / M \quad (1)$$

where  $INCR$  is the inconsistency rate,  $INC_i$  is the inconsistency count for pattern  $P_i$ ,  $h$  is the number of patterns, and  $M$  is the total number of instances.

In earlier works such as [1], inconsistency rate is applied into the search algorithms. A threshold  $\delta$  is usually defined at the beginning. For each feature subset  $S$  selected by search algorithm, the inconsistency rate  $INCR$  is calculated. If  $INCR \leq \delta$ , then  $S$  is considered to be consistent. The original threshold  $\delta$  is updated. The process is repeated until a stopping criterion is reached.

We use the consistency rate for algorithmic purposes. Consistency rate is similar to inconsistency rate except that consistency count,  $CC$ , is computed. Instead of subtracting  $N_1$  from  $N$  to get  $INC_i$  for pattern  $P_i$ , we consider  $N_1$  as the consistency count. Thus, the consistency rate can be expressed as follows:

<sup>1</sup> Visit <http://www.wku.edu/~huanjing.wang/SEKE> for the tool of consistency based feature selection framework.

$$CR = \sum_{i=1}^h CC_i / M \quad (2)$$

where CR is consistency rate,  $CC_i$  is consistency count for pattern  $P_i$ ,  $h$  is the number of patterns, and  $M$  is the total number of instances.

From the definitions of inconsistency rate and consistency rate, we can draw the following equation:

$$CR = 1 - INCR \quad (3)$$

Consistency rate has the monotonic property. An evaluation criterion is monotonic if for a dataset  $D$  and a measure CR, there exists feature subsets  $S_i$  and  $S_j$  where  $S_i \subset S_j$ , then  $CR(S_i, D) \leq CR(S_j, D)$  – a proof for this can be found in [1]. Consistency rate is also applied differently in feature selection. At the beginning, we consider the full feature set as the optimal feature set and calculate the consistency rate  $\delta$ . According to the monotonic property, no feature subset that has size less than full feature set can have consistency rate greater than  $\delta$ . For each generated feature subset, if the corresponding consistency rate is equal to  $\delta$  and the size of the feature subset is smaller, the previous best feature subset is replaced. This process continues until it hits a stopping criterion.

## 2.2 Search Algorithms

Search strategies are very important, since a good search strategy can not only reduce the computational cost but also improve the accuracy. The searching process usually focuses on three aspects: where to start, how to produce the next candidate subset, and when to stop. Based on these three aspects, the searching strategies include exhaustive search, heuristic search, probabilistic search, etc [3].

### 2.2.1 Exhaustive Search (ES)

Exhaustive search uses the algorithm to generate every possible combination of feature subset and compute the respective consistency rates. A threshold is set up at the beginning according to the consistency rate calculated for the first selected feature subset which is also set up as best feature subset. As the algorithm proceeds, the current best subset may be replaced by one with same or higher consistency rate and that the new set is smaller. The exhaustive search can start with either a set with one feature and continue by adding features into set or with a full feature set and then remove features from the set.

It is obvious that exhaustive search is time consuming and computationally expensive as it calculates every combination and that many of them may be redundant. The efficiency deteriorates fast with the size of the search space. For example, if a testing dataset has  $n$  features, the number of combinations is  $\sum_{i=1}^n C_n^i$  which implies that there

will be  $\sum_{i=1}^n C_n^i$  times of calculations for determining

consistency rate. The cost for exhaustive search is  $O(2^n)$ , where  $n$  is the number of features in original data set. In conclusion, the exhaustive search is inefficient and costly for a large number of features. An example for the exhaustive search is Focus [1], which is implemented in this paper for comparison purposes.

### 2.2.2 Heuristic Search (HS)

There are two fundamental goals for computing algorithms: finding a way to use less amount of running time and producing an optimal solution. A heuristic algorithm is used when there is no known way to find an optimal solution in which case the goal of the heuristic is to develop a simple process with provable better running time and good solution. Since exhaustive search algorithms take significant amount of unnecessary time and computationally costly, heuristic algorithm is a good alternative to complete quickly and return a decent result.

There are many heuristic search techniques in practice such as Best-First search [14], A\* search [14], Iterative Deepening A\* search [14], SetCover [6], etc. The original idea for SetCover is that two instances with different class labels are said to be “covered” when there exists at least one feature which has different values for the two instances [6]. In other words, two instances with two different class labels are considered to be consistent if two instances have at least one distinctive feature value between them. SetCover is implemented in this paper for comparison purposes.

### 2.2.3 Probabilistic Search (PS)

As mentioned earlier, a traditional feature selection process consists of four parts: feature generation, feature evaluation, stopping criterion and testing. The common goal of feature selection is to find the smallest feature subset with the highest merit based on an evaluation criterion. The search algorithm stops searching when such a feature subset is found. In probabilistic search, stopping criterion can be defined otherwise. Some probabilistic search techniques combine with the heuristic search algorithm to identify the most useless feature during each iteration and thus generate a better candidate subset by eliminating the useless feature. The probabilistic searches are often given a number as a parameter to specify how many times the search is going to run. Each iteration (run) generates a new subset randomly from the remaining features which trimmed off the most useless one during the previous iteration. As the time of the next iteration, the accuracy is achieved at a high computational time.

Other probabilistic search algorithms employ two stopping criteria combining generation time and when the best subset is found. Such combination guarantees accuracy and

the search is stopped not solely based on running times but also the result – therefore avoiding unnecessary lavish expenditure. As the probabilistic search proceeds, the feature subsets are randomly generated with equal probability, once a consistent feature subset is selected that satisfies the threshold the search will stop regardless of the specified running time. In other cases where data may have large number of features and instances, if the general purpose is to find a result with a certain amount of tolerance, setting the running time as the main stopping criterion is the most reasonable method. LVF is a probabilistic search algorithm [8] that is implemented in this paper for comparison purposes.

### 3. PROPOSED SEARCH ALGORITHM

The above feature selection algorithms generate one and only one consistent feature subset. They fall into the category of filter model. We present a new feature selection algorithm called Automatic Hybrid Search (AHS) that will generate at least one consistent feature subset and is a hybrid of the filter model and the wrapper model. A classifier will be used to decide a final feature subset if several consistent feature subsets exists. AHS relies on the monotonic property of consistency rate. This property gives us the following facts:

- (1) The full feature set has the highest consistency rate  $\delta$ . In other words, the consistency rate of any feature subset is less than or equal to  $\delta$ ;
- (2) The superset of a consistent feature subset is also consistent;
- (3) If  $CR(S_i, D) \leq CR(S_j, D)$ , then  $CR(S_i \cap f, D) \leq CR(S_j \cap f, D)$  where  $f$  is a feature not in  $S_i$  and  $S_j$ .

The proposed AHS algorithm uses the above facts and works as follows: the consistency rate of full feature set is computed first and is used as the stopping criterion. Starting from the size one consisting of any feature, consistent feature subsets that have local highest consistency rate are selected. These selected feature subsets will be used to generate supersets. Repeat the process until feature subsets that have the same consistency rate with  $\delta$  or the full feature set is reached. If more than one feature subsets are generated, a classifier (we use C4.5 [7]) will be used to decide which feature subset is selected based on classification error rate. C4.5 is an algorithm for inducing classification rules in the form of a decision tree from a given dataset.

Below we provide the Automatic Hybrid Search (AHS) algorithm, where the  $conCal(S, D)$  function calculates the consistency rate of a given feature set  $S$  for a given dataset  $D$  and the  $combinationSet(list, length)$  function generates every possible combinations according to the list and length passed as parameters.

**Algorithm:** Automatic Hybrid Search Algorithm.

#### Step 1:

Input:

- D, dataset;
- S, full feature set of D.

Output:

$L$ , consistent feature subsets.

- (1)  $L = S$ ;
- (2)  $\delta = conCal(S, D)$ ;
- (3)  $T =$  all subset  $S'$  in  $S$  where  $|S'| = 1$ ;
- (4)  $max = -\infty$ ;
- (5) While the size of any set in  $T < |S|$  {
- (6)  $tempSet = \phi$ ;
- (7) for each set  $T'$  in  $T$  {
- (8)  $tempCal = conCal(T', D)$ ;
- (9) if ( $max < tempCal$ ) then {
- (10)  $max = tempCal$ ;
- (11)  $tempSet = \phi$ ;
- (12) add  $T'$  to  $tempSet$ ;
- (13) }
- (14) if ( $max = tempCal$ ) then
- (15) add  $T'$  to  $tempSet$ ;
- (16) }
- (17) if ( $max \geq \delta$ ) then {
- (18)  $L = tempSet$ ;
- (19) return  $L$ ;
- (20) }
- (21) else if  $|tempSet| = |T|$  then {
- (22)  $T = combinationSet(T, size + 1)$ ;
- (23) }
- (24) else {
- (25) for any set  $tempSet'$  in  $tempSet$
- (26) append  $tempSet'$  with  $f$  where  $f$  is any feature in  $S$ , not in  $tempSet'$ ;
- (27)  $T = tempSet$ ;
- (28) }
- (29) }
- (30) return  $L$ ;

#### Step 2:

Input:

- $L$ , consistent feature subsets from step 1.

Output:

$T$ , selected feature subset.

- (1)  $min = \infty$ ;
- (2)  $T = \phi$ ;
- (3) for each feature subset  $L'$  in  $L$  {
- (4) calculate error rate  $r$  using C4.5 with  $L'$ ;
- (5) if ( $r < min$ ) then {
- (6)  $min = r$ ;
- (7)  $T = L'$ ;
- (8) }
- (9) }
- (10) return  $T$ ;

#### 4. EXPERIMENTAL RESULTS

To compare the performance of AHS to the other feature selection algorithms, we used the Credit Approval dataset [13] and SPECT Heart dataset [13]. The Credit Approval dataset contains credit card application data – the instances that have missing values have been removed. The SPECT Heart dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each patient is classified into two categories: normal and abnormal. The datasets are shown in Table 1.

**Table 1: Experimental datasets**

Name	Number of instances	Number of features
<b>Credit Approval</b>	673	15
<b>SPECT Heart</b>	80	22

The four algorithms were first compared by examining their empirical runtime. Ten runs were done for each feature selection method. Table 2 shows the average run-time over ten trials. We can see the run-time of HS, PS and AHS are much smaller than ES. The runtime for AHS is longer than HS and PS when the original feature set is larger since AHS is a hybrid algorithm and involved learning and selecting feature subset. The execution time is relatively small for all tested algorithms. However, the savings in execution cost of HS, PS and AHS will have significant implications for large scale datasets with tens or hundreds of thousands of features and when the attribute value of a feature is diverse.

**Table 2: Average Run-time comparisons (in milliseconds)**

Feature Selection Methods	Dataset	
	Credit Approval	SPECT Heart
<b>ES</b>	13761219	27110657
<b>HS</b>	2043	1109
<b>PS</b>	4931	226
<b>AHS</b>	4234	24562

A tool to run the feature selection algorithms is developed in Java. Figure 1 shows the AHS result for the SPECT Heart dataset, and where 11 features are selected using AHS. Figure 2 shows the HS result for the SPECT Heart dataset, and where 15 features are selected using HS.

Table 3 shows the number of features selected through each algorithm for different dataset. The result for AHS is same as ES for all dataset. The feature subset selected by AHS is close to the optimal solution.

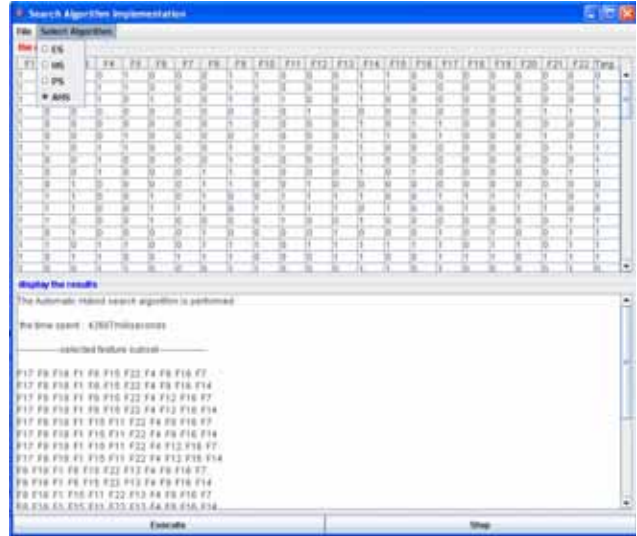


Figure 1: AHS result for SPECT Heart dataset

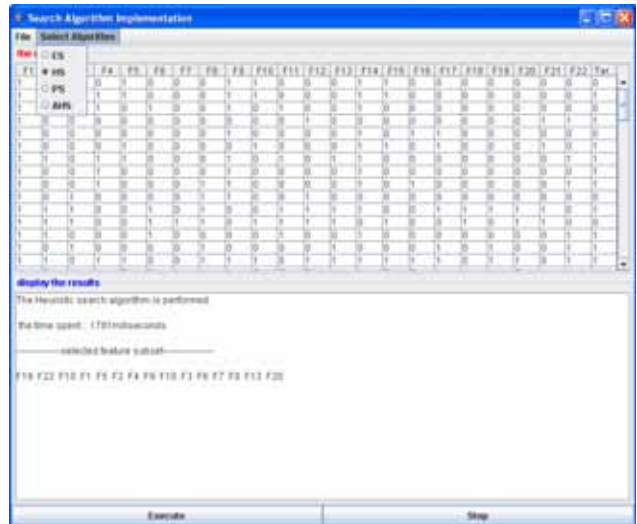


Figure 2: HS result for SPECT Heart dataset

**Table 3: The Size of Consistent feature subset**

Feature Selection Methods	Dataset	
	Credit Approval	SPECT Heart
<b>ES</b>	3	11
<b>HS</b>	3	15
<b>PS</b>	4	15
<b>AHS</b>	3	11

In order to evaluate how our algorithm of feature selection affects classification, we employed the well known classification algorithm C4.5 [7] on the SPECT Heart dataset. The training dataset has 60 instances and test dataset has 20 instances. We used C4.5 as an induction algorithm to evaluate the error rate on selected features for

each feature selection method. Nodes in a C4.5 decision tree correspond to features and the leaves of the tree correspond to classes. The branches in a decision tree correspond to their association rule. Table 4 shows the error rate of the decision tree for each feature selection method. As seen in the experimental results, AHS performed same with HS and PS, but provided a smaller size of feature subset.

**Table 4: Results for the C4.5 Algorithm**

Method	Error rate
ES	21.25%
HS	18.75%
PS	18.75%
AHS	18.75%

## 5. CONCLUSIONS

In this paper, we have reviewed the framework for consistency based feature selection and explained the basic concepts of different feature selection model, i.e., filter, wrapper, and hybrid model. We provided a brief review on an evaluation criterion, the consistency rate measurement. We examined its properties such as monotonic property. Three typical search algorithms, exhaustive, heuristic, and probabilistic search are investigated in this paper. A hybrid search algorithm, called Automatic Hybrid Search (AHS) is proposed.

Instead of stopping on finding only one result, AHS is able to find several consistent feature subsets with the same consistency rate. C4.5 is integrated into the algorithm in order to further ensure that the result is more accurate with a slight trade off of computational time. AHS has been evaluated on the Credit Approval and SPECT Heart datasets. The experiments have shown that AHS performed well at feature selection with relative less runtime cost, yielding a smaller size of selected feature subset, and provided similar or lower error rates than the more traditional approaches such as ES, HS and PS. The findings suggest that AHS is more sensitive to the number of features than to the number of instances in the dataset.

Future work will focus on experimental analysis on more datasets with larger feature spaces. It would be interesting to explore the measures that can allow several related data

mining techniques to work together and facilitate handling different types of data.

## 6. REFERENCES

- [1] M. Dash and H. Liu, "Consistency-based search in feature selection", *Artificial Intelligence*, Vol. 151, Nos. 1-2, p. 151-176, 2003.
- [2] H. Wang, A. Parrish, R. Smith, and S. Vrbsky, "Variable selection and ranking for analyzing automobile traffic accident data", *Proceedings of 20<sup>th</sup> Annual ACM Symposium on Applied Computing*, 2005.
- [3] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, Issue 4, p. 491 – 502, 2005.
- [4] R. Kohavi and G. John, "The wrapper approach", *Feature Selection for Knowledge Discovery and Data Mining*, pp. 33–50, Kluwer Academic Publishers, New York (1998).
- [5] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research*, 3:1157-1182, 2003.
- [6] A. Oliveira and A. Vincentelli, "Constructive induction using a non-greedy strategy for feature selection", *Proceedings of Ninth International Conference on Machine Learning*, p. 355-360, 1992.
- [7] J. R. Quinlan, *C4.5: programs for machine learning*, Los Altos, California: Morgan Kaufmann, 1993.
- [8] H. Liu and R. Setiono, "Feature selection and classification – A probabilistic wrapper approach", *Proceedings of Ninth International Conference on Industrial and Engineering Applications of AI and ES*, p. 419-424, 1996.
- [9] M. Berry and G. Linoff, *Data Mining Techniques for Marketing, Sales, and Customer Support*, John Wiley & Sons, 1997.
- [10] M. Lenzerini, "Data Integration: A Theoretical Perspective", *PODS 2002*: 233-246.
- [11] S. Zhang, C. Zhang, and Q. Yang, "Data preparation for data mining", *Applied Artificial Intelligence*, 17:375–381, 2003.
- [12] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, 1, 131-156 (1997).
- [13] *UCI Repository of Machine Learning Databases*, <http://archive.ics.uci.edu/ml/>.
- [14] B. Coppin, *Artificial Intelligence Illuminated*, Jones and Bartlett, 2004.

# Improving Text Document Clustering by Exploiting Open Web Directory

Gaurav Ruhela and P.Krishna Reddy

Center for Data Engineering

International Institute of Information Technology (IIIT-H)

Hyderabad, India

ruhela\_gaurav@research.iiit.ac.in and pkrreddy@mail.iiit.ac.in

*Abstract—The process of term extraction and weighting affects the performance of information retrieval, search engines and text mining systems. A text document is abstracted as a vector of terms, and the weight for each term is usually given by using popular TF-IDF method. In the TF-IDF method, the weight of a term is a function of its frequency in the document and in overall document collection. The similarity computation by cosine similarity method is influenced by common terms (and their weight) between two document vectors and ignores the semantic relation between terms. We can use the generalization property of hierarchical knowledge repositories to establish that the terms correspond to specific instances of some generalized term. These generalized terms can be used to enrich the document vector, by enriching and weighting we intend to obtain better similarity values between two documents. In this paper, we have proposed an improved term extraction and weighting method by exploiting the contextual/semantic relationship between terms using knowledge repositories such as open web directories. The experiment results show that the proposed approach improves clustering performance over other term extraction and weighting approaches.*

## I. INTRODUCTION

The vector space model [1] abstracts a document as a vector which consists of terms (features). Similarity computation between documents in an effective manner is an important aspect for text document clustering. Similarity between two documents depends on highly weighted common terms. So, selection of appropriate terms and computing their weight is a crucial issue as it influences the clustering performance.

Term weight represents the importance of the term for a given document. Terms that do not describe document's content often intend to induce noise and degrade the performance of the system. The goal of term identification and weighting scheme is to effectively distinguish informative terms from the non-informative ones, and to assign more weight to the informative terms. The TF-IDF [2] and other weighting methods consider syntactic similarity and ignore semantic aspects. For example, consider two documents, one with word "BMW" and other with "Jaguar". The cosine similarity method with existing weighting method gives similarity value between two documents as zero. It can be noted that even though these two documents are different, they are semantically related as both are related to car models. So, identifying similarity between two documents requires investigation of efficient approaches to identify semantic relationship between the documents.

In this paper, we intend to improve clustering performance

by enriching and weighting feature vector. We have proposed an improved term extraction and weighting method by exploiting the semantic relationship between the terms using knowledge repositories such as open web directories. We have exploited the notion that any two terms are related if they have been used in the same context. Based on this notion, for any two terms, if we extract corresponding related terms and include them in document vector, the similarity between two vectors will improve, which further improves the cluster quality.

The contribution of this paper is twofold. First, we propose a framework that performs feature generation (using open web directory) and enriches the feature vector with new, more informative and discriminative features. Second, we propose a weighting scheme to weigh generalized terms in topic paths, which assigns more weight to the terms representing the context of the document.

The rest of this paper is organized as follows. In Section II we review related work. In Section III, we explain the proposed term extraction and weighting scheme. We discuss experiment results in Section IV and conclude in section V.

## II. RELATED WORK

Current state of the art term weighting schemes can be categorized into three classes supervised, unsupervised and context based weighting schemes.

Supervised term weighting schemes are based on the distribution of word in different categories. Machine learning techniques and probabilistic approaches are used to enhance learning from available knowledge [3][4]. Efficiency of these models depends on the quality of the sample sets used for training.

Unsupervised term weighting schemes do not use information on membership of training documents. Simplest model in this category is the boolean model based on set theory. Weight of a term  $t_i$  of document  $\vec{d}_x$ ,  $w \in \{0,1\}$ , is given on the basis of absence or presence of a term. TF-IDF [2] and its various variants assign non binary weight to terms according to their importance for a particular document. Several weighting schemes like LTU [5] and INQUERY [6] are introduced which take use of document length as well. However the importance of capturing context is not considered.



Regarding context based approach, word sense disambiguation is an active topic of research which focuses on finding the correct sense in which the word has been used [7][8][9]. To capture the context of the document, there have been efforts to augment features from resources like “Yahoo Web Directories”, “Wikipedia”, “Wordnet” etc. Work that uses web directories to gain information about the context of the document is discussed in [10], [11][12]. Yahoo categories in [11] were used as knowledge source to classify the web pages into Yahoo categories. In [12], intentions in dialogues of instant messaging applications are captured, which are used for advertising. Question answering system [13] uses predictive annotation in which a token is added into the query to identify potential answers to questions in text.

In [14], an approach is proposed to enrich document vector with conceptual terms using Wordnet [15]. After enriching the terms, all conceptual terms are considered to be at same generalization level and TF-IDF weighting scheme was suggested to assign weights to both document terms and the terms added from Wordnet. Several thresholds are used to put a limit on the number of words to be added.

In this paper we made an effort to develop an unsupervised term weighting scheme that neither require any tagging of text nor any kind of training process, and test the clustering efficiency with different term-weighted vector representations. The proposed approach differs from previous approaches in many aspects; we introduce a term extraction method using topic paths of open web directory. We assign weights to document terms and conceptual terms differently. We introduce various factors which should be considered while assigning weight to conceptual terms. These factors are described in further sections.

### III. PROPOSED TERM WEIGHTING SCHEME

We first explain the basic idea and then we discuss the proposed approach to enrich and weight the document vector.

#### A. Basic Idea

Cosine similarity between two document vectors  $\vec{d}_1$  and  $\vec{d}_2$  is computed as follows,  $Sim(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{|\vec{d}_1| |\vec{d}_2|}$ , where, ‘.’ indicates the vector dot product and  $|d|$  is the length of document vector  $\vec{d}_d$ . Cosine similarity with traditional term identification and term weighting fails to find similarity between two documents that share a topic, but have different terminology. If we use a knowledge resource such as open web directory in which, a given term relates to a context, and the context, in turn, relates to a collection of terms, then we can extract related terms for each term in the document. In a simple generalization hierarchy of web directory, a term at a higher level is a generalized concept for all the terms under this node e.g. sport is a generalized concept for football, cricket, baseball etc. By adding related terms to the feature vector, two different terms which have the same context may get the same related generalized terms. As a result, there is an opportunity to increase similarity between two documents.

Using hierarchical categories of web directories, it is possible to add additional features to the document vector without their literal occurrence in the document. This enriched feature vector has document terms along with the generalized categorical terms which represent the context of the document. This would increase the similarity between two documents even if they did not had common vocabulary, but were semantically related.

*Example:* Consider two document vectors, one document vector contains the term “BMW” and other document vector contains the term “Jaguar”. If we calculate cosine similarity of these two documents, the similarity value returned would be ‘0’ even though both documents contain information about cars. By exploiting the hierarchical knowledge resource such as open web directories, it is possible to improve the performance of similarity computation. When a term is queried in an open web directory such as DMOZ<sup>1</sup>, it returns several topic paths and respective count value. The topic paths obtained for the terms “BMW” and “Jaguar” are listed in Table I and Table II respectively. In Table I, first topic path for “BMW” has “Makes and Models” as its immediate generalized term followed by “Autos” and then “Recreation”. If generalized terms of “BMW” and “Jaguar” are included in the corresponding document vectors then “Makes and Models”, “Autos” and “Recreation” will be common to both document vectors. As a result, the cosine similarity between these two documents will be greater than ‘0’.

So, there is opportunity to improve the performance of similarity computation by exploiting hierarchical knowledge resources like open web directory.

#### B. Description of Proposed Approach

We explain the proposed approach after explaining the relevant terminology.

- *Document Term (DTerm<sub>i</sub>):* Given a document we extract ‘n’ ( $n > 0$ ) terms to form initial document vector. We call each term as document term (DTerm). The ‘ $i^{th}$ ’ term in the document vector is denoted by  $DTerm_i$  ( $1 \leq i \leq n$ ).
- *Topic Path (TPath<sub>ij</sub>):* When web directory is queried with a  $Dterm_i$ , it returns ‘p’ topic paths. Each topic path contains a sequence of terms. The first term is a  $DTerm_i$  itself and rest terms are generalization of preceding term.  $TPath_{ij}$  is the  $j^{th}$  ( $1 \leq j \leq p$ ) topic path of  $Dterm_i$ . Formally,  $TPath_{ij}$  is defined as follows:

$$TPath_{ij} := \langle x_k : x_{k-1} : \dots : x_0, count \rangle \quad (1)$$

Here,  $x_0$  is  $DTerm_i$ ,  $x_k$  is a immediate generalization of  $x_{k-1}$ , ‘k’ is the level in the topic path and *count* is the number of related web pages which falls under the respective topic path. Table I and Table II shows five topic paths related to words “BMW” and “Jaguar” respectively.

- *Generalized term (GTerm<sub>ijk</sub>):* Given a topic path, the terms other than DTerm are called generalized

<sup>1</sup><http://www.dmoz.org/>

TABLE I  
TOPIC PATHS AND COUNT FOR TERM **BMW**

Link no.	Categorical Link	Count
1	Recreation: Autos: Makes and Models: BMW	91
2	Recreation: Motorcycles: Makes and Models: BMW	90
3	World: Deutsch: Freizeit: Auto: Marken: BMW	69
4	Business: Automotive: Motorcycles: Makes and Models: Retailers: BMW	29
5	Home: Consumer Information: Automobiles: Purchasing: By Make: BMW	11

TABLE II  
TOPIC PATHS AND COUNT FOR TERM **JAGUAR**

Link no.	Categorical Link	Count
1	Recreation: Autos: Makes and Models: Jaguar	67
2	Games: Video Games: Console Platforms: Atari: Jaguar	34
3	Shopping: Vehicles: Parts and Accessories: Makes and Models: European: British: Jaguar	28
4	Kids and Teens: School Time: Science: Living Things: Animals: Mammals: Jaguar	9
5	Sports: Football: American: NFL: Jacksonville Jaguars: Jaguar	4

terms.  $GTerm_{ijk}$  is a generalized term occurring in the  $TPath_{ij}$  for  $k \neq 0$ , where ‘ $k$ ’ is the level number.

- $WDTerm_i$  and  $WGTerm_{ijk}$ : The weights of  $DTerm_i$  and  $GTerm_{ijk}$  are denoted by  $WDTerm_i$  and  $WGTerm_{ijk}$  respectively.

Relation between  $DTerm_i$  and its topic paths is depicted in Fig 1. Here, a term points to its immediate generalized term.

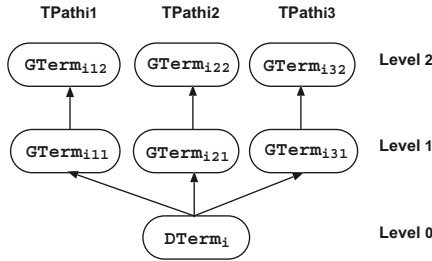


Fig. 1. Relation between  $DTerm_i$  and its topic paths

For each document, proposed approach follows the following steps: (i) Generation of DTerms for document, (ii) Determining weight for the terms in topic path, and (iii) Formation of enriched document vector. Details of these steps are as follows:

1) *Generation of DTerms for document*: Let ‘D’ be the total number of documents in dataset. Feature vector of each document  $d_d$  ( $1 \leq d \leq D$ ) in ‘n’ dimensional term space is  $\vec{d}_d = (t_1, t_2 \dots t_n)$ . In more general form document  $\vec{d}_d$  is a vector of weights.  $W_d = (w_{1d}, w_{2d} \dots w_{nd})$ , where  $w_{id}$  ( $1 \leq i \leq n$ ), is weight of term  $t_i$  of document  $\vec{d}_d$ . High frequency words (Stop-words) such as ‘i’, ‘the’, ‘am’, ‘and’ etc, are removed using a stop-word list. Then the terms are reduced to their basic stem by applying a stemming algorithm. As of now low frequency terms are kept, but later we conduct experiments by removing low frequency terms from the document vector.

2) *Determining weight for the terms in topic path*: Extract TPaths for every DTerm of the document. Each  $TPath_{ij}$  consists of one DTerm and sequence of GTerms. Assign weight for DTerm as well as GTerm using the following

methods.

- *Assigning weight to document terms ( $DTerm_i$ )*  
We can use any weighting scheme to weigh DTerms. Here, we use TF-IDF, effects of weighting with TF, LTU and INQUERY are also shown in experiment section. Same weights (WDTerm) will be used to weigh GTerms in next sub-section.
- *Assigning weight to generalized terms ( $GTerm_{ijk}$ )*  
GTerms that are overlapping for several terms should receive higher weights than the GTerms that appear in isolation from the others. But, if only frequency criterion is considered, terms at the high level will get more weight, being generalized term of many terms their frequency will be high. So, along with the addition of generalized terms, their weighting is also important. The weight of a generalized term is a function of three factors namely (i) GTerms of important terms are important, (ii) Importance of topic path, and (iii) Less weight to more generalized terms. The weight of  $GTerm_{ijk}$  of  $DTerm_i$  in  $j^{th}$  topic path and  $k^{th}$  level can be formalized as

$$WGTerm_{ijk} = WDTerm_i * imp_j * exp^{-k} \quad (2)$$

We elaborate these three factors one by one.

- *GTerms of important terms are important*  
 $WGTerm_{ijk}$  is proportional to  $WDTerm_i$  because high weight indicates the importance of a term for the document. In other words, terms with high weight represents the document in a more informative manner. So, the generalized terms of high weighted DTerms also become important. Thus the weight of generalized terms should be proportional to the weight of DTerms.
- *Importance of topic path ( $imp_j$ )*  
For a  $Dterm_i$ , several topic paths are obtained because of polysemy nature of term. The importance of a term for different topic paths might differ. The importance of a term towards ‘ $j^{th}$ ’ topic path is captured by the probability of a term to occur in

that topic path, i.e.  $imp_j = \frac{count(j)}{\sum_{m=1}^p (count(m))}$ . High  $imp_j$  shows that a term is used more frequently in ' $j^{th}$ ' topic path and thus more related to it. Thus the GTerms occurring in this topic path are important.

- *Less weight to more generalized terms ( $exp^{-k}$ )*  
GTerms closer to DTerm represents the document relatively more precisely than the other more generalized terms in the topic path. So, GTerms which are close to the DTerm in topic path should get relatively more weight than the GTerms which are farther away in the topic path. We use a decreasing function (exponential) to assign less weight with the increase in level of topic path. *For Example:* Consider topic path 1 of DTerm “BMW” in Table I, “BMW” is at level ‘0’, “Makes and Models”, “Auto” and “Recreation” are at level ‘1’, ‘2’ and ‘3’ respectively. “Makes and Models” is immediate generalized term and thus defines “BMW” better than the other generalized terms like “Auto” or “Recreation”. Thus more weight should be given to “Makes and Models”.

TABLE III  
GENERATION AND WEIGHTING OF  $EDtermSet_i$

Input: $DTerm_i$
Output: $EDtermSet_i$ , Weighted and Enriched-term Set for $DTerm_i$
1: $WDTerm_i = tf(DTerm_i) * idf(DTerm_i)$
2: $EDtermSet_i = \{DTerm_i, WDTerm_i\}$
3: $TPathList = \{\}$
4: $TPathList = AddTopicPaths(DTerm_i)$ //Add topic paths of $DTerm_i$
5: <b>foreach</b> $TPath_{ij}$ <b>in</b> $TPathList$
6: <b>foreach</b> $GTerm_{ijk}$
7: $imp_j = \frac{count(j)}{\sum_{m=1}^p (count(m))}$
8: $WGTerm_{ijk} = WDTerm_i * imp_j * exp^{-k}$
9: $EDtermSet_i = EDtermSet_i \cup \{GTerm_{ijk}, WGTerm_{ijk}\}$
10: <b>end</b>
11: <b>end</b>

In this step, by giving each  $DTerm_i$  of the document (along-with its weight) as input, the pairs of GTerms along-with their weights are obtained. We define this collection of  $DTerm_i$  and its GTerms from all topic paths as enriched term set ( $EDtermSet_i$ ). Algorithm for  $EDtermSet_i$  generation and weighing its terms is given in Table III. Formally,

$$EDtermSet_i := \{DTerm_i \cup GTerm_{ijk}\} \forall j, k \quad (3)$$

3) *Formation of enriched document vector ( $\vec{d}'_d$ ):* For every term of every  $EDtermSet$ , if a term from  $EDtermSet$  is not present in  $\vec{d}'_d$ , then the term along with weight is added to the  $\vec{d}'_d$  to represent this term. If the term exists in  $\vec{d}'_d$  then the weight of this term is added to its instance in  $\vec{d}'_d$  (algorithm in Table IV). Formally,

$$\vec{d}'_d := \cup\{EDtermSet_i\} \forall i, i \in n \quad (4)$$

TABLE IV  
FORMATION OF ENRICHED DOCUMENT VECTOR  $\vec{d}'_d$

Input: 'n' Enriched-term Sets ( $EDtermSets$ )
Output: Enriched and Weighted Document Vector $\vec{d}'_d$ .
1: $d'_d = \{\}$
2: <b>foreach</b> $EDtermSet_i, i \in n$
3: <b>foreach</b> $term_t$ <b>in</b> $EDtermSet_i$
4: <b>if</b> (AlreadyExists( $term_t$ )) //Check in $d'_d$
5:       IncrementWeight( $term_t$ ) //Weight added to instance in $d'_d$
6: <b>else</b>
7: $d'_d = d'_d \cup \{term_t, Wterm_t\}$ //Add term and its weight to $d'_d$
8: <b>end</b>
9: <b>end</b>

To put more insight into the formation of enriched document vector, we pictorially show enriched document with two DTerms,  $DTerm_x$  and  $DTerm_y$  (Fig. 2). DTerms along-with their topic paths are merged to form enriched document. For term  $DTerm_x$ ; three topic paths are obtained  $TPath_{x1} := \langle A_{x13} : A_{x12} : A_{x11} : DTerm_x \rangle$ ,  $TPath_{x2} := \langle B_{x22} : B_{x21} : DTerm_x \rangle$ ,  $TPath_{x3} := \langle C_{x33} : C_{x32} : C_{x31} : DTerm_x \rangle$ . While term  $DTerm_y$  has two topic paths  $TPath_{y1} := \langle B_{y12} : B_{y11} : DTerm_y \rangle$ ,  $TPath_{y2} := \langle C_{y23} : C_{y22} : C_{y21} : DTerm_y \rangle$ .

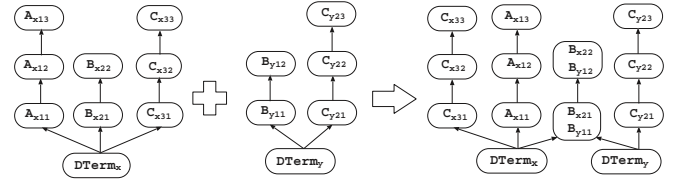


Fig. 2. Document after merging all topic paths for all DTerms

Suppose,  $B_{x21}$  and  $B_{y11}$ ,  $B_{x22}$  and  $B_{y12}$  are same terms, thus, after merging kept in same node of enriched document. Weight of these terms is the summation of their weight from both topic paths.

Factors which affect the weight of node having term  $B_{x21}$  are: (i) importance of paths  $DTerm_x \rightarrow B_{x21}$  and  $DTerm_y \rightarrow B_{y11}$ , (ii) weight of  $DTerm_x$  and  $DTerm_y$ , (iii) distance of  $B_{x21}$ ,  $B_{y11}$  from  $DTerm_x$  and  $DTerm_y$  respectively. So, final weight of a GTerm will be summation of exponentially decreased weight from all the DTerms occurring in the document, whose topic path this GTerm appears in.

#### IV. EXPERIMENTS

We conducted experiments on WebData<sup>2</sup> dataset consisting of 314 web documents already classified into 10 categories. To measure the effectiveness of different weighting schemes, we cluster the documents with Bi-Secting-KMeans [16], a variant of KMeans. Several runs of Bi-Secting-KMeans are used to register the average purity value. Cosine similarity is used as

<sup>2</sup><http://pami.uwaterloo.ca/~hammouda/webdata/>

proximity measure. For evaluation of cluster quality we use the following purity measure.

Let the given test clusters be  $C = \{C_1, C_2 \dots C_{10}\}$  and clusters obtained by several approaches be  $C' = \{C'_1, C'_2 \dots C'_{10}\}$ . Each resulting cluster  $C'_i$  from a partitioning  $C'$  of the overall document set  $D$  is treated as if it were the result of a query.

The precision of a cluster  $C'_i \in C'$  for a given category  $C_j \in C$  is given by  $Precision(C'_i, C_j) = \frac{|C'_i \cap C_j|}{|C'_i|}$ .

The overall value of purity is computed by taking the weighted average of maximal precision values:

$$Purity(C', C) = \sum_{C'_i \in C'} \frac{|C'_i|}{|D|} \max_{C_j \in C} Precision(C'_i, C_j) \quad (5)$$

TABLE V

TERM WEIGHTING SCHEMA.  $tf$  MEANS TERM FREQUENCY,  $D$  IS THE TOTAL NUMBER OF DOCUMENTS IN COLLECTION,  $df$  IS THE DOCUMENT FREQUENCY,  $dl$  IS THE DOCUMENT LENGTH,  $avg\_dl$  IS THE AVERAGE DOCUMENT LENGTH FOR A COLLECTION.

Name	Term Weight Schema
TF	$tf$
TF-IDF	$tf * \log(\frac{D}{df})$
LTU	$\frac{(\log(tf) + 1) * \log(\frac{D}{df})}{0.8 + 0.2 \frac{dl}{avg\_dl}}$
INQUERY	$\frac{tf}{tf + 0.5 + 1.5 \frac{dl}{avg\_dl}} \frac{\log(\frac{D + 0.5}{df})}{\log(D + 1)}$

We have conducted experiments with three type of feature vectors:

- Only document vector (*ODV*): In this experimental setting, vectors of original documents ( $\vec{d}_d$ ) are used for document-document similarity, that is, similarities are calculated without adding GTerms. Feature vector is then weighted with the weighting schemes in Table V.
- Enriched document vector with common weighting (*EDVCW*): Here, GTerms are added to the document vector  $\vec{d}_d$ . Enriched document vector is concatenation of document terms and all generalized terms,  $\vec{d}'_d = \{\vec{d}_d, \vec{G}_d\}$ . Same weighting scheme is used to weight both  $\vec{d}_d$  and  $\vec{G}_d$ , for instance, if TF-IDF is used to weight  $\vec{d}_d$ , TF-IDF will be used to weight  $\vec{G}_d$  too.
- Enriched document vector and proposed weighting (*EDVPW*): Enriched document vector,  $\vec{d}'_d = \{\vec{d}_d, \vec{G}_d\}$ . What differs from *EDVCW* is the way to assign weight to  $\vec{G}_d$ . We consider each weighting scheme mentioned in

Table V for DTerms and use proposed approach to assign weights to GTerms.

#### A. Experiment without pruning terms

In this experiment we have not used any threshold to remove non-informative words. In Table VI and Fig 3, we can see that EDVPW outperforms other feature vector representations for all the weighting schemes. INQUERY being the exception got less purity than ODV by a small margin.

TABLE VI  
AVERAGE PURITY VALUES WITHOUT PRUNING TERMS

Feature Vector	TF	TF-IDF	LTU	INQUERY
ODV	0.7053	0.7580	0.6920	<b>0.6565</b>
EDVCW	0.5231	0.6883	0.6121	0.6367
EDVPW	<b>0.7480</b>	<b>0.7730</b>	<b>0.7282</b>	0.6503

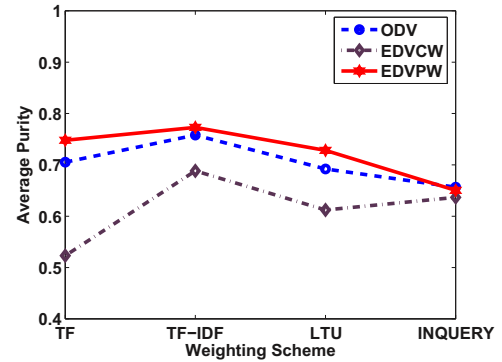


Fig. 3. Average Purity values without Pruning Terms

It can be observed that, surprisingly, the clustering performance did not improve with the addition of generalized terms and weighing them with the same weighting scheme used for DTerms (EDVCW vector). On the other hand it got degraded. With the addition of highly generalized terms (which tend to be super concept of several terms) without appropriate weights, discriminating power between two documents got crippled. As a result, proximity computation between two documents is effected, thus the quality of clusters.

#### B. Experiment with pruning terms

In this experiment, we have used a simple pruning method by removing terms which have less than certain term frequency and carried out two experiments. One is by selecting the document terms having term frequency greater than 6 and then adding generalized terms for each document term (Fig 4) and Table VII. Similarly, we have carried out another experiment by selecting the terms having frequency greater than 31 (Fig 5) and Table VIII.

The experiments conducted by pruning terms showed better results over without pruning approach. Low frequency terms do not represent the document, so the addition of their generalization terms adds noise to the document vector. Due to this noise, similarity values and thus the purity values were compromised.

TABLE VII  
AVERAGE PURITY VALUES AFTER REMOVING TERMS WITH  
FREQUENCY < 6

Feature Vector	TF	TF-IDF	LTU	INQUERY
ODV	0.7169	0.7460	0.6908	0.6417
EDVCW	0.5188	0.6755	0.6139	0.5883
EDVPW	<b>0.7379</b>	<b>0.7952</b>	<b>0.715</b>	<b>0.674</b>

TABLE VIII  
AVERAGE PURITY VALUES AFTER REMOVING TERMS WITH  
FREQUENCY < 31

Feature Vector	TF	TF-IDF	LTU	INQUERY
ODV	0.7190	0.7343	0.6919	0.6908
EDVCW	0.5128	0.6513	0.5904	0.6114
EDVPW	<b>0.7393</b>	<b>0.7681</b>	<b>0.7332</b>	<b>0.7168</b>

It can be observed that proposed approach improves performance of other weighting schemes. From the results, we can conclude that, each document vector should be prepared as follows: (i) Enrich document with GTerms, and (ii) Follow TF-IDF (TF, LTU, INQUERY) to weigh DTerms and proposed approach to weigh GTerms.

## V. CONCLUSION AND FUTURE WORK

In this paper, we have presented a context based unsupervised term extraction and weighting scheme. We have exploited the notion that any two terms are related if they have been used in the same context. In the proposed approach, a given document is enriched with generalized terms using open web directory. We have proposed a term weighting scheme to give appropriate weights to both document terms and generalized terms. One of the factors to weigh GTerms is the importance of the topic paths. So, GTerms with high weight in enriched document vector represents the context of overall document, thus disambiguating the context of terms. The performance results show that the proposed weighting scheme gives better clustering performance over existing weighting schemes. As part of future work, we are planning to conduct detailed experiments by considering other types of datasets. In addition, we are planning to conduct experiments by applying dimension reduction techniques like latent semantic analysis.

## ACKNOWLEDGMENT

The work was carried out while the authors are working in eSagu project at IIIT, Hyderabad, India. The authors are thankful to Media Lab Asia for its support.

## REFERENCES

- [1] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, 1958.
- [2] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Ithaca, NY, USA, Tech. Rep., 1987.
- [3] H. Xu and C. Li, "A novel term weighting scheme for automated text categorization," in *ISDA '07: Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 759–764.
- [4] M. Lan, C. L. Tan, and H. B. Low, "Proposing a new term weighting scheme for text categorization," Boston, 2006, pp. 763–768.
- [5] C. Buckley, "New retrieval approaches using smart: Trec 4," 1996, pp. 25–48.

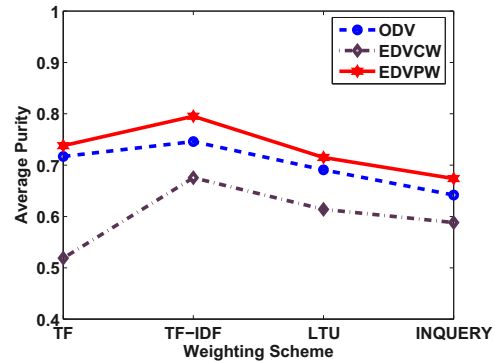


Fig. 4. Average Purity values after removing terms with frequency < 6

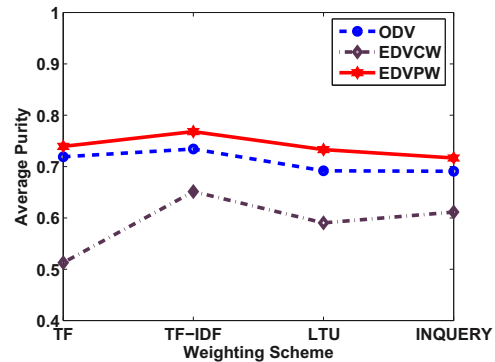


Fig. 5. Average Purity values after removing terms with frequency < 31

- [6] A. U. Kini, "On the effect of inquiry term-weighting scheme on approved by: Query-sensitive similarity measures," 2005.
- [7] E. Agirre and G. Rigau, "Word sense disambiguation using conceptual density," in *In Proceedings of the 16th International Conference on Computational Linguistics*, 1996, pp. 16–22.
- [8] Y. Karov and S. Edelman, "Similarity-based word sense disambiguation," *Comput. Linguist.*, vol. 24, no. 1, pp. 41–59, 1998.
- [9] E. Gabrilovich and S. Markovitch, "Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization," *J. Mach. Learn. Res.*, vol. 8, pp. 2297–2345, 2007.
- [10] Y. Labrou and T. Finin, "Yahoo! as an ontology: using yahoo! categories to describe documents," in *In Proceedings of the 8th International Conference On Information Knowledge Management (CIKM)*, 1999, pp. 180–187.
- [11] Mladenic and Dunja, "Turning yahoo to automatic web-page classifier," in *European Conference on Artificial Intelligence*, 1998, pp. 473–474.
- [12] H.-C. Huang, M.-S. Lin, and H.-H. Chen, "Analysis of intention in dialogues using category trees and its application to advertisement recommendation," in *Proceedings of the Third International Joint Conference on Natural Language Processing*, Hyderabad, Andhra Pradesh, India, 2008, pp. 625–630.
- [13] J. Prager, E. Brown, A. Coden, and D. Radev, "Question-answering by predictive annotation," in *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2000, pp. 184–191.
- [14] A. Hotho, S. Staab, and G. Stumme, "Wordnet improves text document clustering," in *In Proc. of the SIGIR 2003 Semantic Web Workshop*, 2003, pp. 541–544.
- [15] G. A. Miller, "Wordnet: a lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [16] S. M. Savaresi and D. L. Boley, "On the performance of bisecting k-means and pddp," in *Proceedings of the First SIAM International Conference on Data Mining (ICDM-2001)*, 2001, pp. 1–14.

# AUTOMATED NURSING KNOWLEDGE MANAGEMENT USING INDEXING

<sup>†</sup> Shihong Huang, <sup>†</sup>Sucharita Chinchankar, <sup>†</sup>Abhijit Pandya, <sup>†</sup>Sam Hsu, <sup>\*</sup>Marilyn Parker

<sup>†</sup>Department of Computer Science & Engineering

<sup>\*</sup>Christine E. Lynn College of Nursing

Florida Atlantic University

Boca Raton, FL U.S.A.

(Shihong, schincha, pandya, samh, mparker)@fau.edu

## Abstract

*Properly and efficiently capturing and managing nursing knowledge is essential to advocating health promotion and illness prevention. This paper proposes a document-indexing framework for automating classification of nursing knowledge based on nursing theory and practice model. The documents defining the numerous categories in nursing care model are structured with the help of expert nurse practitioners and professionals. These documents are indexed and used as a benchmark for the process of automatic mapping of each expression in the assessment form of a patient to the corresponding category in the nursing theory model. As an illustration of the proposed methodology, a prototype application is developed using the Latent Semantic Indexing (LSI) technique. The prototype application is tested in a nursing practice environment to validate the accuracy of the proposed algorithm. The simulation results are also compared with an application using Lucene indexing technique that internally uses modified vector space model for indexing.*

**Keywords:** knowledge management, LSI, natural language processing, classification, indexing, nursing care.

## 1. Introduction

Nursing is considered as an altruistic profession and the care given by a nurse to the patient is implied as holistic healthcare. A nurse considers the physical, emotional, social, economic, and spiritual needs of the patient. As a result, the care provided by the nurse provides an effective healing alternative that complements that provided by medical doctors. But despite of this tremendous contribution of nurses to the quality of the healthcare, the work done by a nurse remains concealed. So in order to identify and analyze nurses' interventions in improving quality of healthcare, a standardized nursing language is introduced to describe the care that is provided by nurses [2]. These standards are backed up by the various nursing theories. Nursing documentation provides the basis for nurses to communicate with each other and with the rest of healthcare communities. The

data collected by the nurse in assessment period is referred to as nursing knowledge, as it represents the knowledge of the nurse about the patient. With the help of this nursing knowledge and the pre-existing nursing theories, the nurse structures a holistic care plan for the patient. This care plan helps the nurse to provide the patient with the best care needed and help in his speedy recovery. To efficiently and properly capture and manage the nursing knowledge, automated or semi-automated analysis and classification of nursing knowledge are needed. One of the benefits of this automated process of nursing knowledge is to integrate nurses' contribution and nursing caring aspects into the electronic medical records (EMR).

The next section discusses some of the fundamental aspects of nursing knowledge management and representative methodologies for classifying and indexing information. Section 3 presents the research goals and the approach to automate the management and classification of nursing knowledge based on nursing theories. Section 4 illustrates the approach by presenting a prototype software system and testing it in a nursing environment. The simulation results and analysis are presented. Finally, Section 5 summarizes the research and outlines future work.

## 2. Background and Related Work

Nursing knowledge is often captured in natural language and in textual format. Proper capturing and managing this knowledge is essential for nurses to provide a proper diagnosis and treatment plan. This section presents the background of some general approaches to nursing knowledge management, and describes representative methodologies for natural language classification.

### 2.1. General Approach to Nursing Knowledge Management

One of the methods used to capture and analyze the nursing knowledge is use of printed-paper forms. These forms outline categories of the nursing theory models along with the series of check boxes and empty spaces where nurse can manually enter the expressions from the assessment data and map it to the respective category.

This method is both time consuming as well as error prone as it depends largely on an individual who analyzes the data. The same data when analyzed by multiple nurses will produce inconsistent results. Also storing of this data in paper form is associated with all kind of storage issues. Some computer aided software engineering tools are available to semi-automate this process of analyzing the nursing knowledge data.

## **2.2. Existing CASE Tool Support**

Atlas/ti is a software tool used for the qualitative analysis of large bodies of textual, graphical and audio/video data. It has multiple functions to administer, extract, compare and aggregate the meaningful data from a collection of data [9]. In order to use Atlas/ti to classify nursing knowledge, the text file containing the patients assessment data is linked to the project in Atlas/ti called Hermeneutic Unit (HU). Then each expression in the file is analyzed by a nurse practitioner to map it to a related category from the nursing theory model. The data can then be queried, sorted or represented in a diagram.

This approach addresses the storage issues by storing the data electronically and time consumption issues to some extent as once mapped; the data can then be queried, sorted or analyzed using multiple electronic tools. But one major concern using this software for such purpose is preserving the consistency of the data. In order to achieve the required consistency, this category assignment needs to be automated.

## **2.3. Related Methodologies Used for Information Retrieval**

Among the several information retrieval technologies available today, the following technologies are predominantly used for qualitative data analysis.

### **2.3.1. Semantic Web Approach**

Semantic Web is a framework that makes it possible to store the data in machine understandable format. Semantic web uses URI (Uniform Resource Identifier) to uniquely identify each resource. To make the most use of data, the documents are described using XML (eXtensible Markup Language). XML is the official recommendation of W3C (World Wide Web Consortium) that allows the use of self-descriptive tags to describe the data [10]. This technology needs the documents to be in XML format. As the nursing knowledge is written in plain natural language text, we need to seek a technology that can process the data in the form of natural language text.

### **2.3.2. Naïve Bayes Text Classification**

Bayesian network is a graphical representation that considers probabilistic relationship for variables of interest [11]. Bayesian network classifier obtains network structure and conditional probability table by learning

training data set, and then implements the classification by using Bayesian theorem to compute posterior probability [12].

This technology is widely used for qualitative analyses of data but its accuracy depends on the amount of data used as a reference to classify. As the amount of reference data increases, the classification is more accurate.

### **2.3.3. Latent Semantic Indexing**

Latent Semantic Indexing (LSI) uses statistically derived conceptual indices to match the query and retrieve the information from a set of documents. A truncated Singular Vector Decomposition (SVD) is calculated to predict the structure of word usage in the document. This technique could be used on natural text and can provide accurate results even with the small amount of reference data available. Hence we will exploit this technique to automate the classification of the nursing knowledge.

## **3. A Framework of Nursing Knowledge Management**

This section presents a framework that automatically classifies the nursing data that is in the textual assessment form and mapping it to the corresponding category in the nursing knowledge model based on nursing theories. The architecture of this framework is based on the canonical activities of reverse engineering [22]. The overall process can be divided into three logical steps: Data gathering, knowledge management and information exploration [22]. The following sections describe these three steps in detail.

### **3.1. Data Gathering**

Gathering data from the patient is the first and most essential step. The raw data gathered is the narrative written in natural language by nurse. This data is collected by nurse during the patient's assessment period. We call this data as nursing knowledge as it represents the nurse's interpretation of the patient's condition, his history and his concerns. This data is crucial for nurses to understand the patient's needs and concerns [13].

### **3.2. Knowledge Management**

The nursing model is comprised of number of categories. In order to classify the nursing knowledge into these categories, we need to collect all the information pertaining to each category into a separate benchmark document. The information could be a thorough definition of the category, all the terms that best describe the category along with the example expressions. These benchmark documents are then indexed using Latent Semantic Indexing technique that assigns indices based on the semantics of the document. As the nursing knowledge is expressed in natural text, different terms could be used to express the same concept (synonyms).

Hence by using LSI technique we can ensure the correct classification the data even if the exact matching term is not present in the benchmark document. The documents are then ranked with respect to each sentence in the source document. The documents that are semantically close to the source sentence in the source document are ranked higher. So each sentence in the source document gets mapped to the category represented by the highest ranked benchmark document.

### 3.3. Information Exploration

Information exploration is essential for the nurse to make the efficient use of the structured data. The tree structure representing the structured layout of the nursing knowledge will be presented to the nurse. This makes it easier for her to navigate through the data. Also as the nursing knowledge is sorted and grouped into the different categories in the nursing model, it will facilitate the clear understanding and quick planning.

## 4. Case Study

To validate the proposed framework, the automated classification of nursing knowledge is simulated using Latent Semantic Indexing technique. The benchmark model used in the case study is “Parker/Barry community nurse practice model” [2]. The following sections describe the case study and discuss the simulation results and analysis.

### 4.1. Data Collection

Parker/Barry Community Nurse Practice Model is comprised of 4 main concepts: Caring, Wholeness, Connections and Respect. These concepts are also known as the nursing instrumental values. The benchmark documents are arranged such that each document represents a category in Parker/Barry community nurse practice model. Each document is well structured to contain a thorough definition of the category and example expressions for the respective category. These documents will serve as the benchmark to classify the nursing knowledge. This collection of document will then act as a benchmark to help the further automation.

### 4.2. Knowledge Management Using LSI

This part is the crux of the automation, as it comprises the logic to automate the classification. The creation of the term document matrix includes the following steps. First, stop words are eliminated from the benchmark documents. This process will purge all the words that do not carry any information. Then the stemming algorithm is applied to the document to obtain a list of non-repeating unique terms in the collection of document [16], designated as the universal terms. Weight of each term is calculated by iterating through the universal terms and the

term-document matrix is prepared. Once the term document matrix is created, it is decomposed into three different matrices using singular vector decomposition (SVD) method [21].

To prepare the query matrix Q, a query vector q is generated by matching the concerned query expression that is the nursing knowledge, to the list of universal terms. If the term is present in the query expression, we will set the corresponding element in query vector otherwise we reset the element.

By calculating the cosine similarity between the two vectors, one can find the benchmark document that is most similar to the query expression.

After calculating the cosine similarities between the query matrix and each document matrix, we will have j different ranks for j benchmark documents against each query expression. This process is known as ranking of documents. The document with the highest rank is most similar to the query expression. So the query expression is assigned to the category represented by that benchmark document.

### 4.3. Simulation Results and Analysis

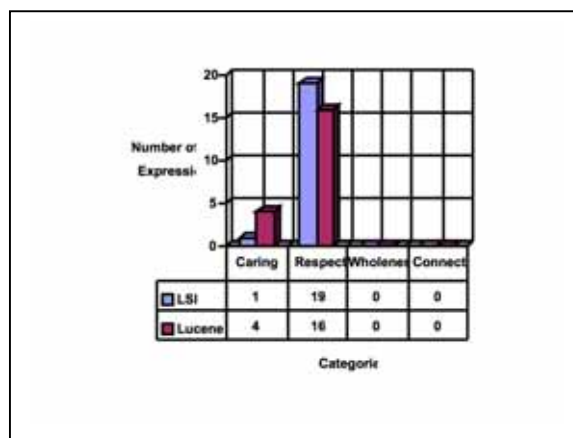


Figure 1: Comparison of results for classification of 20

To compare the results, another application is developed that will use Lucene to index the benchmark documents. Lucene uses a modified vector space model for its search and not Latent Semantic Indexing [17] [18] [19]. The results of Lucene-based application are compared to the application that uses LSI. Figure 1 shows the comparison of results for classification of 20 expressions belonging to ‘Respect’ category. As shown in the figure, 19 expressions were classified correctly using LSI technique as opposed to 16 expressions using Lucene indexing technique. The sentence that was misclassified to belong to the caring category is: “My nurse asked about my daily needs I am vegetarian so I need special meals.” This



sentence could belong to both caring as well as respect category. So it could be mapped to both the categories. This feature of mapping an expression to multiple categories is not implemented in this application and could be added in the future.

The results for 20 expressions of caring category were also compared. For both the applications 16 expressions out of 20 were classified correctly. Hence the average accuracy for application using LSI and Lucene indexing techniques comes up to be 87.5% and 80% respectively.

## 5. Summary and Future Work

The nursing knowledge captured during the interaction between nurses and patients, further with patients' families, is an invaluable part of providing a care plan and monitoring recovery progress. This paper proposes a framework that automatically manages nursing knowledge and maps nursing practice to the caring categories according to nursing theory. To illustrate the validation of the framework, a case study in a nursing practice environment is presented, and the classification results are analyzed and compared with alternative approach. The result comparison shows that the LSI strategy gives 87.5% accurate results compared to the Lucene indexing technique that gives 80% accuracy. Both indexing methods maintain 100% consistency in the results.

To further automate the process of preparing the care plan for the patients, this application can be extended to allow a nurse to enter the action plan related to each classified expression to generate and save the care plan for the patient. This application can even be used to monitor the progress of a patient throughout his treatment. Some special categories could be defined to track the concerns of the patient. Then the application could be modified to generate a graph showing the recovery progress of the patient based on his past and present concerns. Often times, in addition to the medicines prescribed, the care provided by the nurses can help in the speedy recovery of the patient. The analyses of the care plan specified by the nurse on particular visit of the patient and the related graph of patient's recovery progress will further help nurses to understand the factors that helped the patient in his speedy recovery.

## 6. References

[1] Swan, B.A., Lang, N.M., and McGinley, A.M.: "Access to quality care: Links between evidence, nursing language, and informatics", *NURSING ECONOMICS*, November-December 2004, 22, (6), pp. 5

[2] Parker, M.E., *Nursing Theories and Nursing Practice* vol. 2: Philadelphia F.A. Davis Company, 2005.

[3] Meleis, A.I., *Theoretical nursing: Development and progress*. Philadelphia: Lippincott Williams & Wilkins, 1997.

[4] Dorothy, M.C.: "Perspectives of pure science", *Nursing Research*, 1968, 17, (6), pp. 497 - 501

[5] Dickoff, J., James, P., and Wiedenbach, E.: "Theory in a practice discipline: Part I. Practice oriented theory", *Nursing Research*, 1968, 17, (5), pp. 415 - 434

[6] Chinn, P.L., and Kramer, M.K., *Integrated Knowledge Development in Nursing*. St Louis: Mosby, 2004.

[7] Gray J. & Forsstorm, S.: "Generating theory for practice: the reflective technique", in Gray J. & Pratt, R. (Ed.): *Towards a discipline of nursing* (Melbourne: Churchill Livingstone, 1991)

[8] Delaune, S.C., and Ladner, P.K., *Fundamentals of Nursing: Standards and Practices*. Albany, NY: Thomsom Delmar Learning, 2002.

[9] <http://www.atlasti.com/aboutUs.html>, accessed on Sept 2008

[10] Palmer, S.B.: "The Semantic Web: An Introduction", 2001

[11] Chengiu Liu; Wechsler, H.: "A unified Bayesian framework for face recognition". *Proc. IEEE Signal Processing Society 1998 International Conference on Image Processing 1998* pp. Pages

[12] Tang, B.C.Q.L.Z.: "A Clustering Based Bayesian Network Classifier". *Proc. IEEE International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)* Aug 2007 pp. Pages

[13] Foltz, P.: "Using Latent Semantic Indexing for Information Filtering", in Editor (Ed.): *Using Latent Semantic Indexing for Information Filtering* (1990, edn.), pp. 40-47

[14] Mitra, V., Wang, C.-J., and Banerjee, S.: "A Neuro-SVM Model for Text Classification using Latent Semantic Indexing", in Editor (Ed.): *A Neuro-SVM Model for Text Classification using Latent Semantic Indexing* (2005, edn.), pp. 564-569

[15] Sun, J.-T., Chen, Z., Zeng, H.-J., Lu, Y.-C., Shi, C.-Y., and Ma, W.-Y.: "Supervised Latent Semantic Indexing For Document Categorization", in Editor (Ed.): *Supervised Latent Semantic Indexing For Document Categorization* (2004, edn.), pp.

[16] <http://en.wikipedia.org/wiki/Stemming>, accessed Oct 2008

[17] <http://lucene.apache.org/java/docs/>, accessed Oct 2008

[18] <http://en.wikipedia.org/wiki/Lucene>, accessed Oct 2008

[19] Erik, H., and Otis, G., *Lucene in Action* (In Action series): Manning Publications Co., 2004.

[20] Brian, T.B., Garrison, W.C., and Richard, K.B.: "Latent semantic indexing is an optimal special case of multidimensional scaling". *Proc. Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, Copenhagen, Denmark 1992

[21] <http://www.miiisita.com/information-retrieval-tutorial/svd-lsi-tutorial-1-understanding.html>, accessed Oct 2008

[22] Tilley, Scott. "The Canonical Activities of Reverse Engineering" *Annuals of Software Engineering*, Volume 9, Number 1-4, Springer Netherlands, 2000.

# Classifying Web Robots by K-means Clustering

Derek Doran and Swapna S. Gokhale  
Dept. of Computer Science and Engineering  
Univ. of Connecticut, Storrs, CT 06269  
{derek.doran,ssg}@engr.uconn.edu

## Abstract

*Sophisticated Web robots, sporting a variety of functionality and unique traffic characteristics, constitute a significant percentage of request and bandwidth volume serviced by a Web server. To adequately prepare Web servers for this continuous rise in Web robots, it is necessary to gain deeper insights into their traffic properties. In this paper, we propose to classify Web robots according to their workload characteristics, using K-means clustering as the underlying partitioning technique. We demonstrate how our approach can allow an examination of Web robot traffic from new perspectives by applying it to classify Web robots extracted from a year-long server log collected from the Univ. of Connecticut School of Engineering domain.*

## 1. Introduction and Motivation

It has been traditionally believed that the traffic seen by Web servers is from human visitors, which exhibits known and well-studied properties [8, 13]. Recent studies, however, suggest that an increasing proportion of Web server traffic can be attributed to Web robots. Generally, Web robots are autonomous agents that visit a Web site with the purpose of indexing available resources and their location on the Web for search engines [2]. With the evolution of Web 2.0 technologies and the transition towards a semantic Web where autonomous agents visit Web servers on behalf of humans [1], the sophistication of Web robots is expected to rise, and this will inevitably lead to an increase in the volume and intensity of their traffic. Our recent results from the analysis of robot traffic on Web servers at the University of Connecticut (UConn) School of Engineering (SoE) between February 2007 and January 2008, when compared to the results from a study conducted during the 2001-2002 period, confirms this increasing trend. In our study, 18.49% of all requests

were from robots and these contributed to about 7.85% bandwidth consumption, while in the earlier study traffic from major search engine robots<sup>1</sup> represented 8.51% of all requests served which contributed to about 0.65% of all bytes transferred [3].

With an unmistakable trend in increasing robot traffic, Web servers must be adequately prepared to handle such traffic. A critical first step towards such preparation is to gain a deeper understanding of this traffic. Specific efforts to understand robot traffic are essential for two reasons. First, due to the fundamental differences in the way humans and robots crawl Web sites, our understanding of human traffic patterns does not automatically transcend to the crawling behavior of Web robots. Second, modern Web robots sport a wide variety of functionality that dictate their traffic properties including the request and bandwidth volume [4]. The most recent study analyzing robot traffic [3] offers limited insights because it: (i) focuses only on traffic from search engine crawlers; (ii) was performed before the advent of Web technologies that encourage upload of new information by Web users (due to which the robot traffic was less demanding); and (iii) was from an era where advanced robots with specialized functionality were not prevalent.

To understand modern Web robot traffic, composed of robots with varying functionality, design, and visiting intentions, it is first necessary to partition these robots into meaningful groups to highlight their commonalities and to identify their differences. Such classification should consider the behavior of Web robots from many perspectives, including their intended function, workload characteristics, and the types of resources they request. While our earlier research focused on the functional classification of Web robots [4], the objective of this paper is to improve our understanding of robot traffic by classifying these robots according to the workload characteristics they exhibit on

---

<sup>1</sup>Search engine robots were the most dominant type of robots that crawled the Web during the 2001-2002 period.

a server. We demonstrate the feasibility of using K-means clustering for this purpose, by applying it to robots extracted from UConn SoE server access logs. We conclude with a discussion of the new perspectives that such cluster-based classification of robot traffic provides.

The layout of the paper is as follows: Section 2 provides an overview of K-means clustering. Section 3 describes the data along with preliminary analysis. Section 4 applies the K-means clustering to the robot data and discusses the results. Section 5 introduces related work. Section 6 concludes the paper with directions for future research.

## 2. Overview of K-means Clustering

In this section, we present an overview of the K-means clustering technique in the context of the robot partitioning problem. It is a common algorithm that has been used to analyze and partition data in many different domains [6, 14, 5]. We choose K-means clustering to partition Web robots because of its recent success in analyzing Web server requests [8, 9].

To cluster Web robots, it is necessary to define an appropriate distance metric between data points. The selected metric must factor in the likely correlation between observations used to characterize robot traffic; for example the volume of http requests and number of bytes transferred may be correlated [8]. Furthermore, it should also consider that the observations may be measured across different scales; for example inter-arrival times between requests may be measured in seconds, and the average number of requests sent per session, could be measured as a count. We use the Mahalanobis distance, which incorporates both of these considerations [8] to cluster robots. Let an observation of  $n$  features be recorded in an  $n \times 1$  column vector. Then the Mahalanobis distance between two observation vectors  $\vec{x}$  and  $\vec{y}$  is defined as:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}$$

where  $\Sigma$  is the covariance matrix for all observations and the superscript  $T$  denotes the transpose.

K-means clustering requires that the number of clusters  $k$  be selected before clustering commences. Each application of the algorithm is guaranteed to have  $k$  clusters, so different values of  $k$  will lead to a unique clustering result. Thus, the value of  $k$  governs the quality of clustering, making its selection crucial. Because our objective is to partition Web robots so that all the robots in a group will display similar crawling characteristics, we consider two important criteria in select-

ing the value of  $k$ . The first criterion is concerned with minimizing distance within clusters (intra-cluster distance) while maximizing the distance between clusters (inter-cluster distance). Intra-cluster distance is defined as the distance from a vector to the centroid of the cluster to which it is assigned, while inter-cluster distance is defined as the distance from a vector to another one that does not belong to its cluster. Intuitively, the best clustering will be one that maximizes the inter-cluster distance and minimizes the intra-cluster distance. We measure the first criterion using the silhouette coefficient [12] metric, defined as follows: let  $\hat{C} = \{C_1, C_2, \dots, C_k\}$  be the result of a clustering, fully partitioning a set of data points  $D$ . Define the distance of a data point  $d \in D$  to some cluster  $C_i \in \hat{C}$  as

$$dist(d, C_i) = \frac{\sum_{d_i \in C_i} dm(d, d_i)}{|C_i|}$$

where  $dm$  is the distance function between points. Let

$$\alpha(d) = dist(d, C_i^*), d \in C_i^*$$

be the distance from  $d$  to its assigned cluster  $C_i^*$  (i.e. measuring intra-cluster distance) and

$$\beta(d) = \min_{C_i \in \hat{C}, C_i \neq C_i^*} dist(d, C_i)$$

be the distance from  $d$  to the nearest cluster  $d$  is not assigned to (i.e. measuring inter-cluster distance). The *silhouette* of  $d$  is defined as:

$$\phi(d) = \frac{\beta(d) - \alpha(d)}{\max(\beta(d), \alpha(d))}$$

$\phi(d)$  will approach  $-1$  as the inter-cluster distance decreases and intra-cluster distance increases, and will approach  $1$  in the mirroring case. Thus, the closer  $\phi(d)$  is to  $1$ , the better the cluster assignment for  $d$  is. The silhouette coefficient of a clustering is simply the average value of the measure for each data point  $d$ :

$$SC_{\hat{C}} = \frac{\sum_{d \in D} \phi(d)}{|D|}$$

Previous studies suggest that values of  $SC_{\hat{C}}$  greater than  $0.7$  achieve superior separation between clusters, while maintaining data points close to their assigned cluster centroid [7]. Values between  $0.5$  and  $0.7$  are also acceptable, indicating that the data points are sufficiently close to their cluster centroid while still maintaining separation between other clusters.

The second criteria is the degree to which robots are evenly distributed into  $k$  clusters. An even distribution will provide precise insights into the traffic characteristics of robots by clear differentiation. In contrast,

lumping a majority robots into few clusters will lead to general conclusions without any distinctive insights. To measure our second criterion we consider the size of each cluster. In a desirable distribution of robots into clusters, the variance in the size of the clusters must be low, signifying that the robots are not overly concentrated into a single cluster.

We examined both the measures because a high value of  $SC_{\hat{C}}$  does not imply that the cluster size variance will low. A superior choice for  $k$ , for example, may be one where its value of  $SC_{\hat{c}}$  is within an acceptable range and its cluster size variance is smallest. Once the data are partitioned into  $k$  clusters, each cluster is given a unique label  $C_{a,b,\dots}$ , where each subscript is assigned an integer value according to the rank of the cluster’s centroid position in nondecreasing order for each respective traffic feature. This cluster labeling allows the scheme to be easily expandable to consider any number of data features.

### 3. Data Description

The data comprised of 169 robots extracted from a year-long access log from the UConn SoE Web server over the period February 2007 to January 2008. We extracted these robots using a custom log analyzer written in Java that compares the user-agent field from each HTTP request against a database of regular expressions representing well-known Web robots. For each robot we then extracted three metrics: (i) volume of HTTP requests sent, (ii) volume of bandwidth consumed, and (iii) average size of resources requested.

We analyzed the three metrics in a pairwise fashion over the entire set of robots to explore the correlations between them. Figures 1 through Figure 3 show the results of the pairwise analyses of these metrics. In each figure, the top plot includes all data points, while the bottom one focuses in on the most concentrated region to offer a better sense of the data distribution. The top plot of Figure 3 shows a positive linear relationship between bandwidth consumed and volume of http requests, with the correlation coefficient measured at 0.804. This observation matches with previous results suggesting a strong linear correlation between request volume and bandwidth consumption for all server traffic [8]. On the contrary, the average size of requested resources exhibits no observable relationship with both the request volume and bandwidth consumption (Figures 1 and 2), with correlations of 0.035 and  $-0.004$  respectively. These observations thus dispute the belief that a robot, which on average requests very large resources, will also consume a considerable bandwidth or will send a large number of http requests.

The top plots in all the figures indicate that some robots place disproportionate strain on the Web server. Although it is common to filter such outliers before applying clustering, we chose to include them because it is important to understand the traffic from these robots that disproportionately consume server resources from the point of view of server preparation. Furthermore, our limited sample of 169 robots would be pared down further by excluding these outliers.

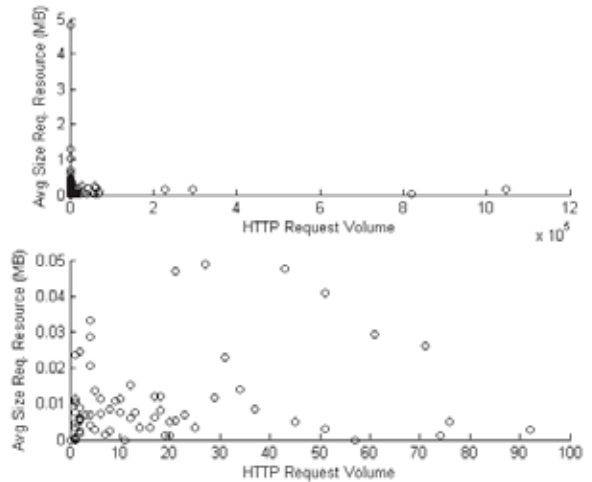


Figure 1. Http request volume vs avg. requested resource size

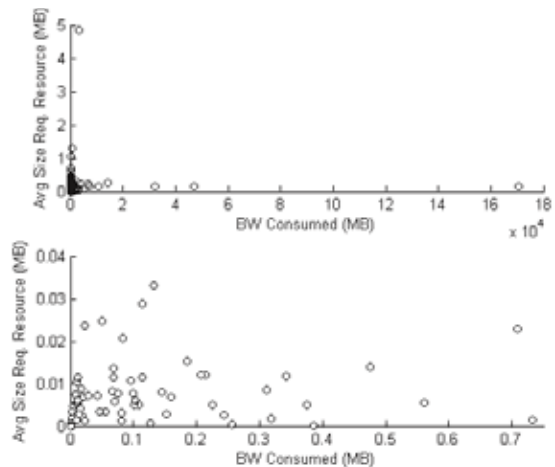


Figure 2. Bandwidth vs avg. requested resource size

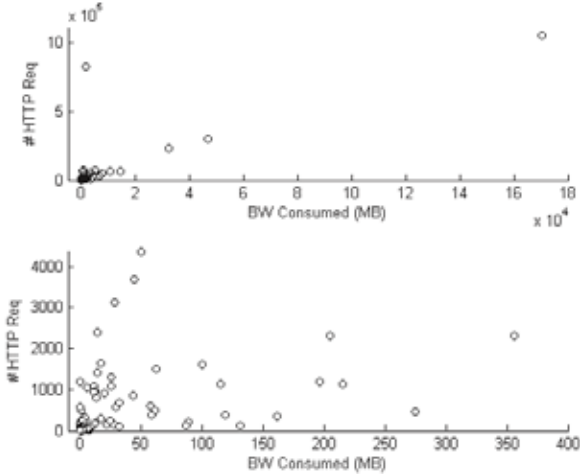


Figure 3. Http request volume vs. bandwidth

## 4. Results and Discussion

We performed K-means clustering with the three metrics for each of the 169 robots. We chose these three metrics to illustrate the feasibility of using clustering to partition Web robots; in practice any number of additional traffic metrics could be used to generate a higher-dimensional clustering. The clustering algorithm was implemented in MATLAB, and verified using several manually-generated test sets that contained clear groupings of the data points. In this section, we first discuss our analysis to select the appropriate number of clusters. Subsequently, we comment on the quality and characteristics of the clusters and the important insights they provide into robot traffic.

### 4.1 Parameter Configuration

To select an appropriate number of clusters that maximizes the silhouette coefficient and minimizes the variance in cluster size, we performed K-means clustering with randomly selected initial centroids for  $k$  ranging from 2 to 12. We limited the maximum number of clusters to 12 due to the small number of robots.

Figure 4 charts the value of  $SC_{\hat{C}}$  as a function of  $k$ . While  $k = 2, 3,$  and  $4$  show very high values of  $SC_{\hat{C}}$ , using so few clusters would offer little insights since this would not appropriately classify the outliers across any metric into its own group. A noticeable dip in the measure is seen when  $k = 5$ , followed by a steady increase until another peak at  $k = 7$  where  $SC_{\hat{C}} = 0.7038$ . For  $7 \leq k \leq 11$ , the levels of the silhouette coefficient indicate a good tradeoff between inter and intra-cluster distances.

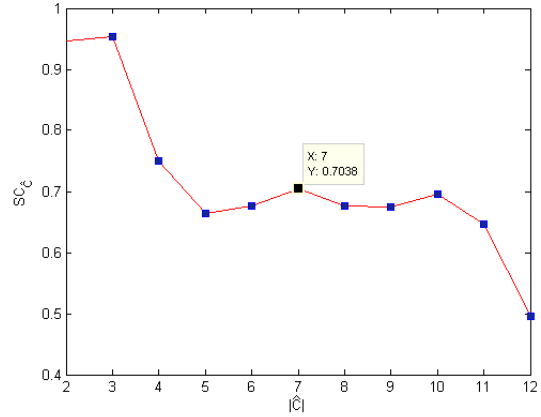


Figure 4. Silhouette coefficients for each k-clustering

Figure 5 charts the variance in cluster size for the same range of  $k$ . When  $k = 12$  the variance in size of each cluster is smallest, however, the respective value of  $SC_{\hat{C}}$  drops significantly. For  $7 \leq k \leq 11$ , the variance in cluster size is small and does not drop significantly as  $k$  increases. Recognizing a peak in the value of  $SC_{\hat{C}}$  and relatively low variance for  $k = 10$ , we choose to partition these robots into 10 clusters.

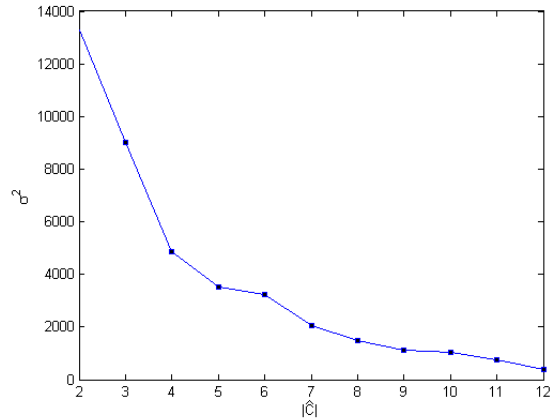
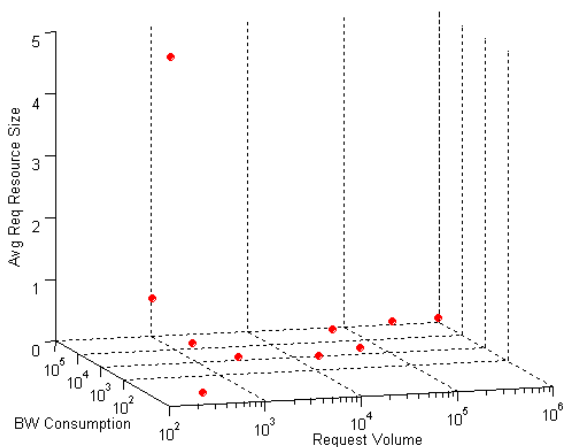


Figure 5. Variance of size of each cluster for each k-clustering

### 4.2 Cluster Characteristics

Table 1 show the average values of each metric or the coordinates of the centroid for each cluster. The clusters are assigned a label  $C_{a,b,c}$  where  $a, b$

and  $c$  represent the cluster rank based on request volume, bandwidth consumption, and average size of requested resource respectively. Figure 6 presents a three-dimensional plot of the positions of cluster centroids, with a log scale for request volume and bandwidth, and a linear scale for the average requested resource size. The figure shows that the centroids are positioned along the request volume and bandwidth axis according to the positive linear correlation observed between these metrics. The centroid positions along the average requested resource size axis, however, are concentrated because robots tend to request very small resources on average [3]. This is especially true for this academic Web server, which is likely to host a large collection of small files.



**Figure 6. Centroid positions for each cluster**

Table 1 also defines size and the boundaries for each cluster across the three metrics. The table reveals that over 63% robots fall into cluster  $C_{2,1,1}$ , whose label suggests that this group of robots request a relatively small volume of http requests, consume little bandwidth and request the smallest resources on average. The membership of this cluster is significantly high due to the presence of outliers, which are forced into their own cluster (for example,  $C_{4,7,10}$  and  $C_{1,3,9}$ ). Because these outliers cannot be ignored, we can accommodate them by refining very large partitions through repeating the clustering only over robots in these partitions. This will produce a hierarchical structure of clusters where the highest-level ones deliver a broad classification of Web robots while lower-level clusters refine a broad class into a series of more specific ones. For example, Table 1 suggests that robots in  $C_{2,1,1}$  exert low demands on the server. Furthermore, this large  $C_{2,1,1}$  cluster also contains robots that do not retrieve any resources. Thus, it may be desirable to partition this cluster fur-

ther to isolate such “no-demand” robots into their own class. Such refinement of clusters can classify robots at any desired level of granularity.

Since robots in this cluster consume relatively fewer resources, they most likely reflect traffic that does not impose significant strain on the server. The difference in the bounds along each metric is also small, which also indicates that robots in this cluster are heavily concentrated. By comparison, clusters of high-demand robots such as  $C_{7,8,7}$ ,  $C_{9,9,5}$ , and  $C_{10,10,4}$  are very wide and have few members. The few robots in this cluster show extraordinary characteristics, and hence, should be examined more closely to determine if their purpose is in the best interests of the Web server. If the investigation reveals that these robots are from commercial services that provide no benefit to UConn SoE for example, they should be blocked from access.

## 5. Related Research

A number of efforts have studied the traffic characteristics of Web robots with an eye towards detecting such robots. Stassopoulou *et al.* [10] employ a detection framework based on a Bayesian network, while Tan *et al.* [11] perform detection based on the navigational patterns of Web robots. Through a more extensive study of robot traffic, focusing on crawlers that belong to five well-known search engines, Dikaiakos *et al.* [3] gain insights into their crawler behavior as a means for separating human users from robots in access logs.

The above efforts consider aggregate properties of robot traffic. In contrast, the research described in this paper applies data clustering to classify Web robots to gain a more detailed understanding of their specific traffic patterns. This exercise is necessary because modern sophisticated Web robots exhibit a wide variety of functionality and visiting intentions, leading to a significant disparity in their crawling behaviors and demands [4]. A detailed study can form the basis of a scheme to detect and block ill-behaved robots. It can also lead to analytical models of robot workloads, which could be used to assess server performance.

## 6. Conclusions and Future Research

This paper presented a classification scheme for Web robots according to their workload characteristics. The scheme utilizes a clustering technique that is extensible to consider any number of traffic metrics. Furthermore, it is iterative so that robots can be classified to any level of granularity. We illustrated the cluster-based

	Req. Volume			Bytes Transferred (MB)			Avg. Req. Size (MB)			size
	min	max	avg	min	max	avg	min	max	avg	
$C_{9,9,5}$	2.27e05	2.94e05	2.6052e05	32190	46930	39560	0.14166	0.15973	.1507	2
$C_{5,4,6}$	2	15559	2102.5	0.3005	2042.5	335.86	0.08820	0.28305	.1545	22
$C_{3,2,8}$	1	5329	634.64	0.34461	1793.7	233.95	0.30705	0.70805	.4474	11
$C_{2,1,1}$	1	4351	339.82	0	99.69	5.9129	0	0.07244	0.0137	107
$C_{10,10,4}$	8.20e05	1.05e06	9.3356e05	1533.7	1.70e05	85977	0.00187	0.16276	0.0823	2
$C_{7,8,7}$	20395	62538	42766	4340.5	14239	8681.2	0.17126	0.24273	0.2059	5
$C_{6,5,3}$	8347	32351	17853	0.38642	2460.5	818.82	4.27e-05	0.09144	0.0360	12
$C_{4,7,10}$	717	717	717	3466.2	3466.2	3466.2	4.8343	4.8343	4.8343	1
$C_{8,6,2}$	38723	70218	58294	181.9	4928.1	1956.2	0.00306	0.07018	0.0323	5
$C_{1,3,9}$	126	402	264	131.28	523.35	327.32	1.0419	1.3019	1.1719	2

**Table 1. Statistics of Robot Clusters**

classification using robots extracted from recent server access logs from the UConn SoE. We then discussed the new perspectives that these classifications provide into robot traffic. Our future research is concerned with applying clustering using several sophisticated traffic metrics. We also propose to investigate the impact of different clustering algorithms on robot classification.

## References

- [1] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. *Scientific American*, May 2001.
- [2] Sergey Brin and Lawrence Page. “The anatomy of a large-scale hypertextual Web search engine”. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
- [3] Marios D. Dikaiakos, Athena Stassopoulou, and Loizos Papageorgiou. “An investigation of Web crawler behavior: characterization and metrics”. *Computer Communications*, 28(8):880–897, 2005.
- [4] D. Doran and S. Gokhale. “Discovering New Trends in Web Robot Traffic Through Functional Classification”. In *Proc. IEEE International Symposium on Network Computing and Applications*, pages 275–278, Cambridge, MA, 2008.
- [5] A. P. Gasch and M. B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol*, 3(11), October 2002.
- [6] Xiaofeng He, Hongyuan Zha, Chris H.Q. Ding, and Horst D. Simon. Web document clustering using hyperlink structures. *Computational Statistics & Data Analysis*, 41(1):19 – 45, 2002.
- [7] L. Kaufman and P.J. Rousseeuw. “*Finding Groups in Data: An Introduction to Cluster Analysis*”. Wiley, 1990.
- [8] Fengbin Li, Katerina Goseva-Popstojanova, and Arun Ross. “Discovering Web Workload Characteristics through Cluster Analysis”. In *Proc. IEEE International Symposium on Network Computing and Applications*, 2007.
- [9] F. Robinson, A. Apon, D. Brewer, L. Dowdy, D. Hoffman, and B. Lu. “Initial Starting Point Analysis for K-Means Clustering: A Case Study”. In *Proc. of ALAR 2006 Conference on Applied Research in Information Technology*, 2006.
- [10] A. Stassopoulou and M. D. Dikaiakos. “Crawler detection: A Bayesian approach”. In *Proc. Int’l Conference on Internet Surveillance and Protection (ICISP’06)*, pages 16–21, 2006.
- [11] Pang-Ning Tan and Vipin Kumar. “Discovery of Web robot sessions based on their navigational patterns”. *Data Mining and Knowledge Discovery*, 6(1):9–35, 2002.
- [12] Ping-Ning Tan, Michael Steinbach, and Vipin Kumar. “*Introduction to Data Mining*”. Addison-Wesley, 2006.
- [13] Jeffrey Xu Yu, Yuming Ou, Chengqi Zhang, and Shichao Zhang. “Identifying Interesting Customers through Web Log Classification”. *IEEE Intelligent Systems*, 20(3):55–59, 2005.
- [14] Di Zhong and Hongjiang Zhang. Clustering methods for video browsing and annotation. Technical report, In SPIE Conference on Storage and Retrieval for Image and Video Databases, 1997.

# Systematic Risk Assessment and Cost Estimation for Software Problems

Jerry Gao, Maulik Shah, Mihir Shah, Devarshi Vyas, Pushkala Pattabhiraman Kamini Dandapani and Emese Bari  
San Jose State University, Email: [jerrygao@email.sjsu.edu](mailto:jerrygao@email.sjsu.edu) eBay Inc

## ABSTRACT

A software product lifecycle consists of a number of phases, including project planning, analysis, design, implementation and maintenance. It is vital to identify and assess the risks and costs of software bugs (or problems) to reduce the related project costs and risks. However, in the real world, engineers lack of systematic methods to estimate and predict the risks and costs caused by software problems (bugs). This paper presents a systematic risk assessment method and tool to estimate the possibility of occurrence of risks posed by the impact of software problems (or bugs). Meanwhile, this paper also provides a systematic way and tool to help engineers to estimate the costs associated with the existing software problems (or bugs). The presented methods can be useful for project managers to make decisions in project budgeting by concerning processes, project risks, and costs. Moreover, some application examples and case study results are reported for bug risk analysis and cost assessment in a real industry project.

## KEYWORDS

Software Risk Assessment, Software Risk Analysis, Bug Cost Estimation, Problem Risk Analysis, and Problem Cost Estimation.

## 1. Introduction

Today, software product development becomes very complicated due to the increasing complexity and scale of today's software systems. The success of a software project depends on many factors. One of them is how to deal with software problems (or bugs) during a project life-cycle and how to assess their related project risks and costs in the software development process. This has a great impact on software product quality during a product development cycle.

However, there have not been enough research efforts made to help engineers and managers to predict the probability of risks and to assess the project cost of problems (or bugs). In the real world, a software product team always encounters the following questions:

- 1) What is the estimated cost associated with a problem's (or bug's) lifecycle?
- 2) What are the cost implications of not fixing a problem?
- 3) What is the possibility that a project development phase might encounter risks?
- 4) How can probability of risks in a particular phase of project be estimated?
- 5) Why should a risk be addressed?

This paper presents a systematic approach to addressing these issues. It discusses one approach for bug-based project cost estimation, and reports a method for project risk assessment concerning software bugs. The proposed bug-impact cost estimation method estimates the possible cost of not fixing a

product problem (or bug). Using this method, engineers are able to prioritize the bug fixing sequence. Meanwhile, the paper also discusses a project risk assessment method for problems in a product's life cycle. Both techniques use the collected problem information in a bug tracking and management system in an organization. The main contribution of this paper is its formal systematic approaches supporting bug-impact cost assessment and risk assessment. Moreover, the paper discusses two prototyping tools based on these methods, and some case study from a real industry project is reported.

This paper is structured as follows. The next section discusses the background and related work in project risk assessment and bug cost estimation. Section 3 presents a systematic method to estimate the cost impact of a bug. Section 4 provides a systematic approach to assessing the probability of risks for a project release. Section 5 presents case studies and the results of employing these techniques in a software product organization, eBay, Inc. Finally, the concluding remarks and future work are included in Section 5.

## 2. Background and Related Work

### 2.1. Software Bug Analysis and Management

With the increasing complexity of today's software systems and the short and tighten project development schedules, risk analysis becomes a very important task for engineers in a software development lifecycle. As indicated in [6], one type of risk analysis has something to with the problems occurred in a project development process. To effectively assess and evaluate the problems (bugs) related costs and bug-based risk, engineers need a systematic approach and tool.

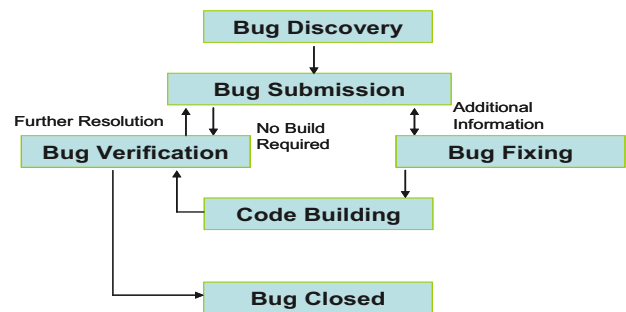


Figure 1 A Bug Processing Workflow

What is a software problem's lifecycle? This refers to the time period from discovery of defect (Status: Open) to confirmation of the defect removal (Status: Closed) constitute the bug lifecycle.

During a software development process, various software bugs (or problems) will be uncovered for a software product. For any



software project team, a software bug (or problem) processing lifecycle will be defined to support bug tracking and analysis. Lots work has been done to support the review of the bug state lifecycle from the post-release perspective to elucidate the various stages in the bug lifecycle. The work presented in [10] is a typical example. Figure 1 shows a typical bug processing workflow and related lifecycle. The non-compliance with the requirement specifications discovered by the client or third party is reported to the software product team. This non-compliance is called a bug and is logged into a bug tracking system with its status as “Open”. In practice, the development team is responsible for the resolution of the bug. Upon resolving the bug, development team changes the status of the bug to “Fixed”. This “Bug-Fix” is tested again and if the same bug reappears, the status of the bug is reset to “Open”. If the bug resolution is confirmed by verification then the status of the bug is changed to “Closed”. Various software product bugs (or problems) can be classified based on their priorities (Fatal, Serious, Minor and Cosmetic) and related service level agreements. The work presented in this paper uses this process workflow as our basis.

Today, as more and more software projects are developed globally. A software bug lifecycle usually spans across different software product development teams. The major questions these teams face with respect to defects (problems) are as follows:

- How to communicate the defect to the development team?
- How to track the lifecycle of the defect?
- How to assign priorities to the defect?
- How to represent the severity of the defect?
- Which defect should be fixed first?
- Can the defect been foreseen before?

There are two perspectives of looking at the questions above [1]. One would be pre-release approach, wherein the emphasis is on the processes involved in defect handling. In this approach, the defect is captured within the organization. The other perspective involves post-release approach. Here the focus shifts from the defect handling processes to the quality paradigms adopted for that product [2]. With software products making inroads into all possible industries, any compromise to the software product quality is not acceptable. Hence, it is vital to fully understand the defect tracking process and the defect lifecycle.

## 2.2. Related Work

Based on our recent literature survey, we found some related works that have contributed to defect tracking and processing. One such well defined defect tracking and cost estimation approach has been put forth by Bala Subramaniam [3] at ISSRe Systems, Inc., in New York. According to him, a well-defined bug tracking and processing workflow is required for Rapid Application Development (RAD) to ensure the compliance of the clients’ requirements at all times. Hence the defect prevention costs, defect appraisal costs, internal and external costs are to be estimated in parallel to RAD model. The Mozilla Foundation has proposed an open source software system to replace in-house bug reporting system by plugging in a comprehensive database management system. This system tracks the bug lifecycle and creates a highly efficient communication and bug handling environment. The salient features of this software system are light weight implementation, quick data

operations and effective ticket tracking system for different priority bugs [3].

In the recent years, there is a growing awareness about potential risks and the alternate solutions to mitigate the effects of risks. Robert W. Ferguson in [4] proposed a normalized risk approach for industry projects. His approach provides clear visibility of the risks to the management. The following risk implication chart shows the different project risk scores for different project releases. Different risk scores are compared and measured at threshold points and appropriate risk mitigation steps are taken. This approach has been tested on multiple projects but not on cross functional locations like out-sourcing centers.

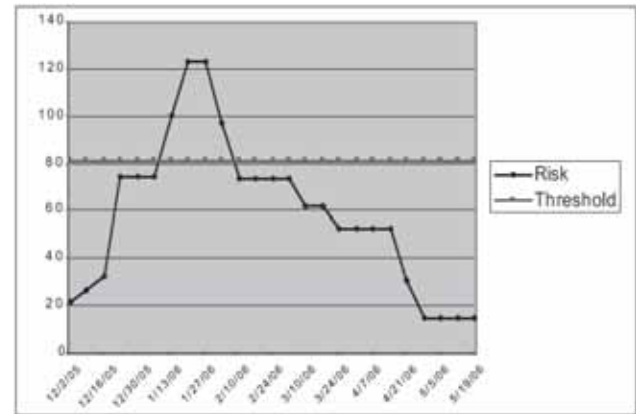


Figure 2 Project risk score [4]

In [9], the paper explores the various factors that impact the cost of not resolving a bug and the factors that induce risks in the completion of a project phase (release). This paper presents an examination into the economics of software quality assurance. An analysis of the software life-cycle is performed to determine where in the cycle the application of quality assurance techniques would be most beneficial. The number and types of errors occurring at various phases of the software life-cycle are estimated. In [6], the authors focus on the software development process and propose a framework for the assessment and management of risk associated with this process. The proposed framework is grounded on a holistic concept termed hierarchical holographic modeling, where more than one perspective or vision of the risk associated with software development is analyzed.

Unlike the existing work, this paper provides a comprehensive approach to calculate the probability of risks and cost impact of bug resolution. The proposed approach is very useful for multi-tier organizations across different functional locations where risks and bugs overlap various tiers and different locations in an organization. Any open bug or potential risk will have immense quality and financial impact in a multi-tier organization. Moreover, this paper reports our implemented prototyping tools for eBay. They provide a systematic way for engineers to unravel the possibilities of project risks and costs of bugs.

In 2007, as a collaborative master project, a group of students at San Jose State University built a bug cost assessment and risk analysis tool for eBay Inc. based on this integrated approach. In

the rest of the sections, we report our cost assessment and risk analysis solution and some case study.

### 3. Bug Cost Estimation

This section presents a systematic method to estimate the cost of bugs, including its impact costs. This method provides a very simple yet powerful technique to calculate the cost of a bug in a product during a software lifecycle. There are many factors that contribute to the cost of a bug. This section first explains our approach and a tool to support bug cost estimation. Next, we present a case study of applying this tool in a real project in eBay.

#### 3.1 Bug Cost Estimation Tool

This section presents a bug cost estimation tool that is implemented as a prototype for eBay Inc. to help engineers to perform cost estimation for bugs in a product release. As shown in Figure 3, the system comprises of four modules:

- Cost assessment user interface – This supports online interactions for engineers to perform bug cost assessment and analysis.
- Communication cost analyzer – This computes the communication costs relating to bug analysis and resolution for a product release.
- Resource cost analyzer – This analyzes the bug-related resource costs for a bug in a product release.
- Impact cost analyzer – This analyzes the bug impact costs of a bug for a product release during a software development cycle.
- Bug information access interface - This supports bug information access between the bug tracking tool in the organization and the bug cost estimation tool.

The rest of the section explains the details of these functional components and their supporting method.

##### Communication Cost Analyzer

This component focuses on the cost incurred due to the process of communication in the lifecycle of a bug (B). This cost is directly reflected when the bug is reported by the customers or third party vendors. The expense incurred due to any email or telephonic communications in the context of the bug discovery are the focus this subsystem. With the knowledge of the number of complaints or emails received and the cost of each complaint, we derived the following equation for communication cost analyzer.

$$\text{Communication Cost}_B = (N_c * C_c)$$

Where  $N_c$  is number of complaints for the bug, and  $C_c$  is the cost of complaints.

##### Resource Cost Analyzer

The Resource Cost Analyzer component focuses on the resource costs incurred due to the bug (B). This subsystem interfaces (through the System Interface) with the existing bug tracking tool or bug database. In a typical bug lifecycle, a resource is assigned to a bug as soon as it is detected and logged into the tracking system. Once the bug is resolved by the Software product team, it is verified and then “CLOSED” else it is again reassigned to a resource. Thus, a bug always has at least one owner in all the stages of its lifecycle until it is “CLOSED”. The resource cost analyzer tracks the resources assigned throughout

bug lifecycle and the number of hours spent by the assigned resources on the bug.

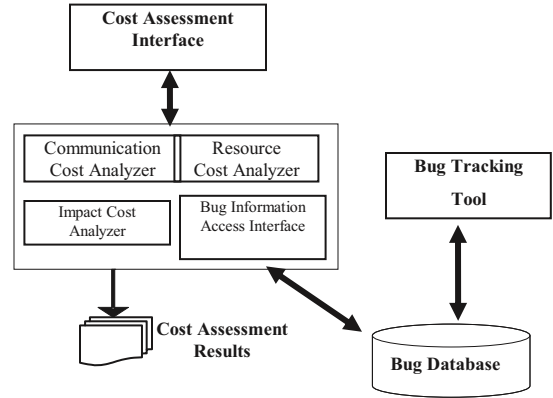


Figure 3 The Bug Cost Assessment Tool

We derived the following equation for computing the resource costs:

$$\text{Resource Cost}_B = \sum (P_i * D_i) \quad (i=1, \dots, n)$$

Where  $P_i$  is the wage (per hour) of the resource working on the bug,  $D_i$  is the total number of hours spent on the bug by an allocated resource. With the Resource Cost Analyzer, engineers can easily compute the resource costs on working on a selected bug on a product line involving different engineering teams.

##### Impact Cost Analyzer

Since each bug is assigned its severity. The bug severity is one of the most crucial attributes of a bug. It represents the impact a bug on the related product. A bug with a higher severity usually has a greater damage and impact to the quality of the product. Every organization usually has a Service Level Agreement (SLA) with its clients. This SLA Agreement defines the maximum time that could be taken to fix a bug based on its severity. Based on the SLA, different priorities are defined for the bugs of a product release. Bugs with higher severities have to be resolved immediately and hence are assigned higher priorities. Thus, the cost impact of a high-priority bug would be greater than that of a low-priority bug. Different levels of bug priorities can be pre-defined as Table 1 below.

Table 1 Bug Priorities and Severities

Bug Priority	Severity
P1	Fatal
P2	Serious
P3	Minor
P4	Cosmetic

Based on the SLA and the priority and average age of a bug (B), we derived the following equation to compute the impact cost incurred due to the violation of a SLA.

$$\text{Impact Cost}_B = ((A_b - SLA) * W_p * C_p)$$

Where,  $A_b$  is the average bug age within the functional domain. It represents the average time taken to resolve bugs of similar priorities and severity in a particular functional domain in an organization. SLA refers to the Service Level Agreement which

indicates the maximum time frame for bug resolution based on a bug's priority.  $W_p$  is the weight of bug priority, and  $C_p$  is the cost of the related priority. Clearly, the higher is the priority, the larger is the weight associated with the priority.

The Impact Cost Analyzer in the tool can be used to assist engineers to compute the impact cost based on the severity of each selected bug in a bug database, which created, tracked, and managed by the existing bug tracking tool.

### Bug Cost Estimator

The Bug Cost Estimator in the tool assists engineers to estimate the total cost of the bug due to factors like communication, resources, bug priorities, and SLA. To predict the total cost incurred for a bug (B) in a product release, we come out the following equation based on the previous computations.

$$\text{Total Cost of bug } B = \text{Impact Cost } B + \text{Resource Cost } B + \text{Communication Cost } B$$

The cost, thus calculated is the cost that an organization would incur if a bug (B) is not resolved. In an environment with numerous bugs with similar severities, calculating the total cost using this formula is a resourceful approach to prioritize bugs. Also, in reality, all risks cannot be resolved effectively; this bug cost estimation technique presents the cost of permitting risks to manifest as bug. Based on the cost estimate of the bug, the risks can be prioritized and resolved. This enables the team to align the resources, time and effort around the cost impact of bug resolution.

### 3.2 A Case Study for Bug Cost Estimation

This case study focuses on the post-release perspective of defects. In the post release scenario, the bugs are found after the completion of a phase. More often, the bugs are discovered by a third-party client or customer. The organizations sign a Service Level Agreement with its customers indicating the maximum time frame for bug resolution and cost impact of non-resolution of bugs within that timeframe. Hence it becomes all the more important to resolve bugs (dependant or independent) within its SLA to avoid its ramifications on cost, resources' effort and time. The expense incurred due to the violation of the Service Level Agreement largely depends on the severity and the priority of the bug. It is vital to track the bug life cycle (using eBay's bug tracking tool), understand the attributes of the bug and present a resolution on time. In the scope of this paper, an issue with PayPal, one of eBay's main functional domains is used for the implementation of the Bug Cost Estimation method. The following section uses various attributes of a bug- **BUGDB00522926** to estimate the cost (in dollars) the bug mathematically.

#### Scenario Analysis for Bug BUGDB00522926

After an auction ends on the eBay website (www.ebay.com), the buyers are allowed to checkout using their PayPal account. Checking out completes the whole process of winning an auction in eBay. Ideally, upon clicking the "Pay Now" button in the eBay website, the user should be able to access his PayPal account. But in reality, upon clicking the "Pay Now" button in the eBay site, the system generates the following error message;

System Temporarily Unavailable. We are unable to locate the information you are requesting. Please try again later." This non-conformance with the requirement specification is logged in as a bug with the bug id, BUGDB00522926. This bug completely sabotages the whole process of winning a bid on the eBay website and hence is critical in nature. This bug spreads across two functional domains eBay marketplace website ([www.ebay.com](http://www.ebay.com)) as well as PayPal. In a scenario with several such bugs detected, an estimate of the cost of the bug resolution would help to align the process, resources and time in a cost effective way. We developed a tool called Bug Cost Estimator to enable the application of our technique in eBay Inc.

Bug Cost Estimation uses the bug information present in the bug database and bug tracking tool. eBay's bug tracking tool and the bug database include the following details about a bug (BUGDB00522926). Let's use **B** to represent this bug. We used the implemented tool to assess the possible cost relating to this bug (B) using the proposed solution.

#### Communication Cost for BUGDB00522926

This module computes the expense incurred by virtue of any communication concerning BUGDB00522926. The primary mode of communication for this bug was via emails and the details of the communications could be obtained from the bug tracker. We found that the actual number of email exchanged to be 455 ( $N_c$ ) and an email complaint costs the organization \$5 ( $C_c$ ). Applying these data to our communication cost analyzer formula, we calculated the actual cost due to communication.

$$\text{Communication Cost } B = (N_c * C_c) = 455 * 5 = \$2275$$

#### Resource Cost for BUGDB00522926

It presents the expense incurred by virtue of the resources working on the bug, BUGDB00522926. Computation of resource cost involves details such as the timestamps for the people working on the bug as well as their hourly wages. From the bug database and bug tracking tool, we inferred the number of resources (here, 3 resources) and the number of hours (here, 5 hours, 6.33 hours and 0.5 hours respectively) they spent on the bug. Based on salary standards, \$40 per hour is assumed as the resource wage/hour.

Using the resource cost analyzer formula, wage per hour ( $P_i$ ) and number of hours ( $D_i$ ) the resource works on the bug, the cost (in dollars) incurred due to the resources can be calculated as follows:

$$\text{Resource Cost } B = (\sum P_i * D_i); \quad (i=1...n) \\ = (40 * 5) + (40 * 6.33) + (40 * 0.5) = \$473.20$$

#### Impact Cost Analyzer

Using impact cost analyzer, we compute the expense incurred due to the violation of SLA. The bug tracking tool provides details such as bug priority (for the BUGDB00522926, priority = P1) and the weight ( $W_p$ ) of the bug (here, weight of the bug BUGDB00522926 is 4). The bug database is used to obtain the information such as statistical average of age ( $A_b$ ) of bugs with similar priority and bug-stage state. The average age of bugs similar to BUGDB00522926, is found to be 72 days. Further, the Service Level Agreement shows that permissible timeframe (SLA) for the resolution of a bug with priority P1 is 22 days.

Finally the total assumed cost of production ( $C_p$ ) for this bug is obtained from the database to be \$1000.

By applying all these data to the formula give below, we can calculate the impact of not resolving the bug within the agreed timeframe.

$$\text{Impact Cost}_B = ((A_B - \text{SLA}) * W_p * C_p) \\ = ((72 - 22) * 4 * 1000) = \$200000$$

Using the computations presented above, the total cost of a bug can be computed as the summation of the three computational results.

$$\text{Total Cost of a bug}_B \\ = \text{Impact Cost}_B + \text{Resource Cost}_B + \text{Communication Cost}_B \\ = \$2275 + \$473.20 + 200000$$

Thus, the estimated cost of a bug amounts to \$202,748.20. This calculation of the cost of a bug is useful in scenario with multiple bugs with similar priorities and is also used in the post-resolution analysis of the bugs.

#### 4. Project Risk Assessment

Every project process and project release has some project risks which usually refer to a certain degree of uncertainty to achieve the success of a project. These uncertainty factors cause project risks in the completion of the project phases. If these risks are not addressed promptly, it might jeopardize the successful delivery of a project release [7]. Hence, it is essential and important to foresee project risks before the completion (or release) of a project phase. This section presents a systematic approach and tool based on mathematical formula to estimate the possibility of risks arising from known actions in a project release. This provided approach numerically quantifies the probability of risks associated with the project phases with a Risk Score. In addition, this section also presents our application example and case study result of using the implemented risk assessment tool.

##### 4.1. The Risk Assessment Tool

As shown in Figure 4, we built a risk assessment tool as a prototype for eBay Inc. to help engineers to perform bug-based risk analysis for a given product release. This risk assessment tool has five components:

- A risk analysis tool interface – This provides a simple graphic user interface to support the interactions with engineers to access different functional components.
- The activity scheduler - This allows engineers to schedule a risk analysis tasks for various bugs stored in a bug tracking tool.
- The bug database access interface – This enables the tool to retrieves bug information from a given bug tracking tool to support the other functional components.
- Risk assessment component – This module allows engineers to perform bug-based risk analysis in a systematic approach.
- Risk migration component – This module keeps and migrate the stored risk analysis results and the related history to support different product release as references for future risk analysis.

The detailed methods supporting these components are described in the following.

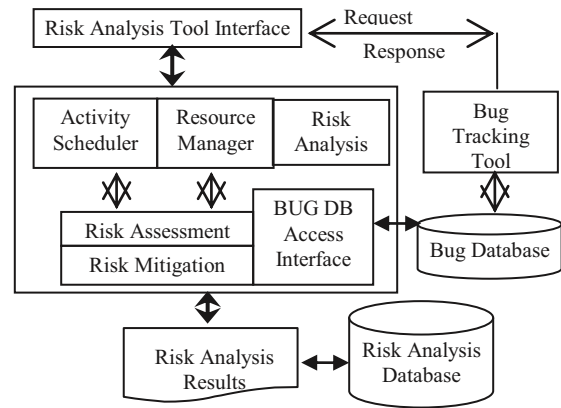


Figure 4. The Risk Assessment Tool

##### Activity Scheduler

The Activity Scheduler identifies the tasks, the sub-tasks and the sequence of tasks in a project phase. It identifies and monitors the schedule (start and end dates) of the project phase and uses the project progression information (in percentage). It also defines main task and subsequent tasks accordingly. All these information could either be obtained from the project database or directly from users using the user interface. Based on all of the information, the tool supports engineers to perform risk analysis functions provided in this tool for associated tasks in a project phase (or release).

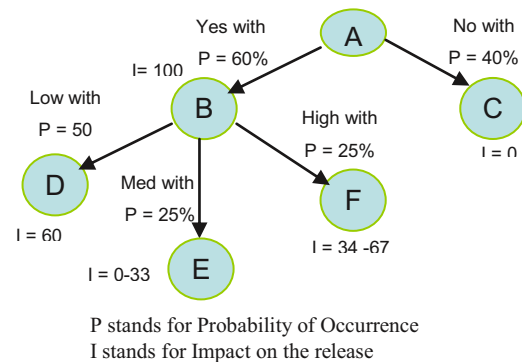


Figure 5 Risk Assessment- Decision Tree

##### Resource Manager

This component identifies all the resources (engineers or managers) who would involve in a selected project phase (or release). It is important to estimate not only the project schedule but also the engineering resources for the project. Similar to the bug cost assessment tool in the previous section, the costs of engineering resources are also obtained or shared in this tool.

##### Risk Assessment

This component assesses the risks based on three factors; namely, users' responses to the project related questions, historical risk analysis data from the repository, and a risk assessment algorithm. The bug related responses are processed using a decision tree algorithm. An example of risk assessment question list is given in Table 3. "P" stands for the probability of the occurrence, and "I" stands for its related impact. To simplify

the processes, we defined these questions for engineers to come out the responses in four ways: (a) Yes/No, (b) True/False, c) a quantitative data in a defined date range. We used the decision-based risk algorithm in [5] to translate the user responses into a risk assessment decision tree. A typical example is shown in Figure 5. Each response becomes a node in the decision tree. Every node is then associated with a probability of occurrence and its impact on the system [5]. A higher probability of an occurrence of a risk indicates that there is a greater chance of encountering the risk in reality. And hence it must be addressed at the earliest. Risks with greater impact indicate that even a single occurrence of the risk will have detrimental effects on the system, as pointed out in [8]. The probability of risk occurrence and impact of the risk on a product release are obtained from the historical data in the risk analysis data repository. Thus, applying the decision tree algorithm and computing the probability and impact of the risk on a product release, we derived the following equation.

$$\text{Risk} = \text{Probability} * \text{Impact}$$

Where, Probability is obtained based on the historical data, and Impact is categorized in the range from 1-3.

**Table 3 Risk Assessment Questions for Engineers**

	Question	Answer	P	I
1	Categorize the release size compared to the other trains of the year	Small	30	30
2	Is this an end of quarter release?	No	15	0
3	Does the release have many big and complex projects or projects with hard dates (committed to the business or legal deadlines)?	Yes	50	70
4	What is the experience of release conductor?	High	30	20
5	Is the release adding new pools and/or new hardware?	No	95	0
6	Are all ROP's completed and approved?	Yes	95	0
7	Percent of projects that completed Dev To QA Handoff	Low	80	75
8	What is the level of confidence that development has completed work in time and all projects have Dev to QA successful?	Low	80	80
9	Is the overall bug finding ratio going down?	Yes	50	18

**Table 2. Risk Types**

Impact Level	Risk Impact type
1	Low
2	Medium
3	High

The formula derived above presents the risk associated with every potential risk causing tasks (activities) in a project phase (or release). A project phase (or release) will comprise of many such tasks. Hence, the Cumulative Risk (CR) in a project release is the summation of the risks of all tasks in that phase. Using the risk information for every activity in the project phase (or release), we derived the mathematical formula for computing the Risk Score of a project phase (or release).

$$\text{Risk Score (\%)} = \text{Cumulative Risk/Maximum Release Score} * 100$$

Where, Cumulative Risk is the summation of the risks of the tasks for a product release (or a project phase). The Maximum

Phase Score is the summation of impacts of all the nodes of the decision tree, i.e., maximum impact of all the tasks in a release.

### Risk Mitigation

Whenever a risk score is generated, it can be saved in a risk analysis database for future use. These scores are used by the Risk Mitigation component to evaluate the alternate plans to mitigate the risks posed in a project phase (or release). The risk score is calculated for each alternative plan. A plan with the least risk score should be selected. This technique practically translates the plans and ideas into quantitative numbers that clearly indicate the percentage of risks in a product lifecycle.

### Risk Analysis Report

This module allows engineers to generate and present a risk analysis report as risk scores in a graphic format for different project phases (or releases). Using this function, engineers can easily identify the stability of a project release. The risk scores are usually high in the early stages of a project phase (or release). Risk scores are definitely a quick and effective indicator of the health of the project phase (or a release).

## 4.2 A Case Study for Risk Assessment

It is important to monitor the progress of the project from its inception to release. In a multi-tier organization (like eBay Inc.) with multiple functional domains, engineering teams work across various geographical locations. Since projects are often bound by time and resource constraint, hence projects may run into lot of risks threatening the successful completion of the project [11]. Often these risks are acknowledged only when they turn into a defect or bug. Here, we report a case study on a project's releases following eBay's bi-weekly release schedule. We look into a new feature released once every two weeks. Its bi-weekly release schedule involves different QA tasks in a development process, including feature testing, code merging and regression testing. To manage the processes effectively, the features are developed by different teams and in different stages. These stages are called Release Seats. Upon developing the feature, every team makes a release. All such releases are collected, merged and tested in a Quality Assurance process in eBay Inc. They involved different teams working across locations, and hence a bi-weekly schedule requires a great deal of insight to avoid issues. In order to handle such short releases effectively, the teams should be equipped with the knowledge about the risks in the release process. Due to the limited scope of this paper, the release schedule strictly begins with the feature testing. The rest of this section presents the application of the Risk Assessment tool and its method onto the release process to quantify risks in its release process.

### Scenario Analysis of Bi-weekly Release

As the first step in a Bi-weekly release phase, the feature development done in the pools are merged in order to be tested in QA environment. The bi-weekly release schedule is logged in an eBay's Project Monitoring tool. It is important to know the schedule of other tasks involved in the releases, such as feature testing, code merging and regression testing. A release conductor or QA manager can use the provided tool to understand the risks associated with each project phase. In his case study, we only examined the project (e575) and its related

risks. The implemented risk assessment tool is used here to assist engineers to analyze and track the associated quality risks associated for each project phase (including internal and external releases).

### Application of Risk Assessment

The Risk Assessment tool estimates the degree of risks (Risk Score) associated with a project release. This tool uses project schedule information, engineers' responses to project related questions in Table 3, These are pre-configured in the tool. They can be grouped into two categories, namely General Questions and Feature Phase Questions. The impact of the current release on all major phases can be concluded from the user's responses to the General Questions. The responses to the Feature Phase Questions provide the specific information for the current release. All these responses are translated into the nodes of a decision tree as shown in Figure 5. The probability of occurrence and impact of occurrence of the each of these nodes are obtained from the statistical data in the risk analysis database. The probability of occurrence of an event in that sequence is indicated by the probability of the node links and the impact of its occurrence is indicated as the impact of the tree nodes. The decision tree can be interpreted into the following table, where P stands for probability of occurrence of a risk posed by the issue in the question.

As discussed in Section 4.1, the risk posed by each of these issues can be calculated by applying probability and impact information in Table 3 below.

$$\text{Risk} = \text{Probability} * \text{Impact}$$

The total risk posed by all the issues in a phase is obtained by summation of risks of the issues in that phase.

#### Cumulative Risk =

$$(0.3 \times 30) + (0.15 \times 0) + (0.5 \times 70) + (0.3 \times 20) + (0.95 \times 0) + (0.95 \times 0) + (0.80 \times 75) + (0.8 \times 80) + 0.5 \times 18 = 39$$

The maximum phase score is calculated by adding the maximum impacts of all the nodes of the decision tree. Applying these data to the formula we derived before, we can calculate the final risk score as follows.

$$\text{Risk Score} = (\text{Cumulative Risk}) / (\text{Maximum Phase Score}) \\ = (39/400) = 9 \%$$

As shown in Figure 6, we computed the risk scores for different phases of the release e575. By analyzing the risk score information, we observed that the risk increases during the early phases and after certain phases, the risk score stabilizes. The Risk Assessor tool saves the Risk Score to the database automatically. We performed 41 iterations of risk assessment on e575 and the results of the same are presented below.

In Figure 6, X legend represents the iteration number and Y represents the risk score for the corresponding iteration. It can be noted the risk score of e575 falls between the range of 25-49, with 25 being lowest and 49 being the highest. This graph is also useful to determine the health of the release phase.

### 5. Conclusion Remarks

This paper presents one systematic approach and tool to estimate bug-related project costs and impacts during a product life-cycle. In addition, the paper provides a systematic solution as a

tool to help engineers and managers to find out the project risks relating to bugs for a product release. Moreover, some application examples and case study results are reported. Using the presented tools, engineers and managers can easily perform bug cost estimation and risk analysis for a project release.

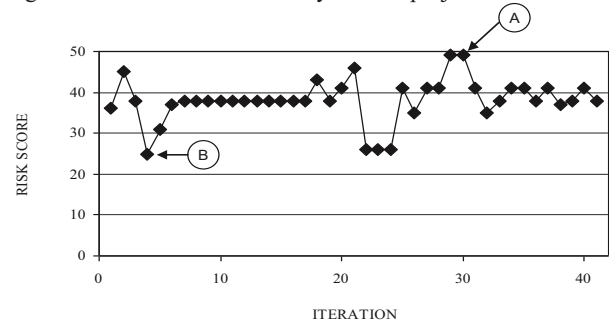


Figure 6 Risk Assessor Graph For Release Phase

The major advantages of these methods are summarized below:

- Eliminates any speculation regarding to the financial impact of bugs and risks.
- Support bug cost estimation and related risk analysis as an effective project and defect management activities.
- Uses the bug information that are tracked and stored by an existing project management and defect management tool.
- Perform bug-related project cost estimation and risk analysis in a systematic and quantitative approach.
- Enables the managers and engineers to make educated decisions based on the cost impact of bugs and risks for each product release in a product development cycle.

### 6. References

- [1] William.T. Ward (1991), "Calculating the real cost of software defects", Hewlett Packard Journal, Oct 1991.
- [2] Rex Black, "Investing in Software Testing: The Cost of Software Quality", 2000. Retrieved from <http://www.stickyminds.com>.
- [3] B. Subramaniam, "Effective Software Defect Tracking - Reducing Project Costs and Enhancing Quality", Crosstalk, April 1999.
- [4] R. W. Ferguson, "A Project Risk Metric", Crosstalk April, 2004.
- [5] P. Robichux, "Essential Guide to Risk Management", Retrieved from <http://www.neverfailgroup.com>.
- [6] C. Chittister and Y. Y. Haimes, "Risk Associated with Software Development: A Holistic Framework for Assessment and Management. Systems, Man and Cybernetics", IEEE Transactions, Vol. 23(3):710-723, 1993.
- [7] Glyn A. Holton, "Defining Risk", Financial Analysts Journal, 60 (6), 2004.
- [8] D. Verndon and G. McGraw, "Risk Analysis in Software Design", IEEE Security & Privacy, Vol. 2, No. 5, 2004.
- [9] M. Davis, "The economics of quality". Proceedings of AFIPS Joint Computer Conference, June 7-10, 1976.
- [10] M. Fischer, M. Pinzger and G. Harald, "Populating a Release History Database from Version Control and Bug Tracking Systems", pp. 23-32, Distributed Systems Group, Vienna University of Technology, Austria, 2003.
- [11] Pertmaster, "Benefit of Project Risk Assessment", Published on 11/1/2006.

# Improving Negotiations through Fuzzy Cognitive Maps

Sergio Assis Rodrigues<sup>1</sup>, Tiago Santos da Silva<sup>2</sup>, Jano Moreira de Souza<sup>1,2</sup>

<sup>1</sup>*COPPE/UFRJ - Computer Science Department, Graduate School of Engineering, Federal University of Rio de Janeiro, Brazil, +55 21 2562-8785, 2562-8676 (fax)*

<sup>2</sup>*DCC-IM/UFRJ - Computer Science Department, Mathematics Institute, Federal University of Rio de Janeiro, Brazil, +55 21 2562-8696, 2562-8676 (fax)*  
*sergio@cos.ufrj.br, tiagoss2005@dcc.ufrj.br, jano@cos.ufrj.br*

## Abstract

*In a decision-making process, a systematic method to manage and measure negotiation aspects provides crucial information to negotiators and has emerged as a key factor to determine agreements' success or failure. Accordingly, this work aims at presenting a negotiation support system which provides functionalities to create negotiation models through settlements' historical information. These models are interpreted as Fuzzy Cognitive Maps (FCM) and the software tests as the convergence of each model. Thus, the system intends to facilitate decision-makers to predict future negotiations behavior and, consequently, improve the chances of an agreement. Besides, this work examines a case study through the lenses of FCM to illustrate real negotiations outcomes and expected value quantifications. The proposal is to compare a model map developed from old negotiations with the results obtained through FCM simulations. As a result, from the use of these mechanisms, negotiators could better visualize alternative ways to improve the chances of successful agreements.*

## 1. Introduction

Negotiations and conflict resolution can be responsible for influencing relationship maintenance and leading institutions towards success or failure. In general, the goal is to reach the planned agreements; however, in decision-making processes, negotiation is directly related to preparation stage and risk assessment. Therefore, the correct management of these uncertain conditions allows one to lead a negotiation in a structured and pro-active way, introducing strategies that may prevent, control and mitigate possible risks that can lead to negotiation failure.

Some specific elements are more discussed in software negotiations, such as scope, time, costs, requisite changes, relationship, interests, administrative issues, contract clauses, power of influence and resources [1][2].

Nevertheless, the challenge is to know how to predict uncertain events and attempt to quantify key negotiation elements in order to prioritize the preponderant ones and, consequently, avoid future problems.

Therefore, initiatives without adequate preparation can unexpectedly lead a promising business to disappointments, specially, among inexperienced professionals, who have natural difficulties to deal with the volume of information to be understood during negotiations. People's cognitive abilities are limited in the simultaneous processing of a high amount of information [3]. As a result, it is the negotiator's responsibility to determine what the most important information to be used on negotiation table is.

In addition, an imperative task for decision-makers is to find a way to predict the deal, although even experienced negotiations can make mistakes in their predictions. Hence, methods to simulate arrangements from initial variables, in which it is possible to manage some aspects over others, represent excellent tools to support deals.

This article attempts to show an approach to test negotiation models in order to improve preparation's stage and also predict deals' variables. Fuzzy Cognitive Maps (FCM) [4][5] are utilized taking into consideration basic negotiation elements to enhance the agreement process.

## 2. Literature Background

In this section, the background description of negotiation methods and FCM are introduced. This work uses these concepts during the proposal system and case study.

### 2.1. Negotiation Process

Overall, negotiation is an activity that requires training, practice, coaching, strategy, and preparation. It allows the

execution of agreements that are mutually acceptable for counterparts, even though different conflicts may occur and external help may be needed [6]. Negotiations can be divided into 4 phases: preparation, value creation, value division and execution [7][8].

Preparation is the most important stage once it provides enough information to facilitate the agreement, defines the issue to be resolved and clearly situates counterparts' interests [9]. Besides, in the preparation step, negotiation should define the ZOPA (Zone of Possible Agreement), or simply zone of potential agreements, which involves the counterparts' satisfaction range [10][11] and concerns the negotiators' expected values. In this zone, some elements can be considered to measure the negotiation's expected value, such as financial values, level of counterpart's relationship, the spending time to reach an agreement, the power of influence and negotiators' strategic interests [2].

In the Value Creation step, it is important to continue exploring the counterpart's interests and generating alternatives that extend mutual gains [12]. At this stage, it is important to avoid criticism and encourage the use of neutrality both to facilitate the relationships and to enable the creation without prior commitments. The Value Division is a step to propose brainstorming on contingent options and to project future agreements. At this stage, neutrality is used to suggest possible ways of distribution and discuss standards and criteria for distributing the generated value [12][13].

Finally, the Execution must establish arrangements to keep track or check adopted decisions and facilitate the commitments maintenance. At this stage, incentives and organizational controls must be aligned and it is essential to work continuously to improve relationships as well as neutrality to resolve disagreements [7][8].

Among these 4 phases, the Preparation step is the most important once it gathers enough information to facilitate the agreement, to define the problem to be solved and to identify clearly the counterparties' interests. Great negotiators have already said that this step is the key to success in negotiations [14][15][16]. Furthermore, it is important to highlight that there are similarities in the best practices used by the major negotiators [17], so it is possible to imagine negotiation models, based on the best practices in specific negotiation scenarios, which supports inexperienced negotiations to become great negotiators.

## 2.2. Fuzzy Cognitive Maps

A Fuzzy Cognitive Map (FCM) is a diagram consisting of nodes and arrows. The nodes represent various qualitative

concepts, while the arrows denote the links between the concepts. Each concept is characterized by a numeric activation value denoting a qualitative measure of the concepts' presence in the conceptual domain. Thus, a high numerical value indicates that the concept is strongly present while a negative or zero value reveals that the concept is not currently active or relevant to the conceptual domain. When a strong positive correlation exists between the current state of a concept and that of another concept in a preceding time-period, it is said that the former positively influences the latter. This relationship is indicated by a positively weighted arrow directed from the causing to the influenced concept. By contrast, when a strong negative correlation exists, it reveals the existence of a negative causal relationship indicated by an arrow charged with a negative weight.

Additional fuzzification to FCMs was introduced via Certainty Neuron Fuzzy Cognitive Maps (CNFCM) [4][18][19], which allow for various activation levels of each concept between the two extreme cases, i.e. activation or not. The updating function of a CNFCM is the following:

$$A_i^{t+1} = f(S_i^t A_i^t) - d_i A_i^t \quad (1)$$

$$S_i^t = \sum_{\substack{j=1 \\ j \neq i}}^n A_j^t w_{ji} \quad (2)$$

In this function,  $A_i$  is the activation level of concept  $C_i$  at some time ( $t+1$ ) or ( $t$ ), equation (2) is the sum of the weighted influences that concept  $C_i$  receives at time step  $t$  from all other concepts,  $d_i$  is a decay factor [19], and (3) is a modified version of the function used for the aggregation of certainty factors [4].

$$f_m(A_i^t, S_i^t) = \begin{cases} A_i^t + S_i^t(1 - A_i^t) = A_i^t + S_i^t - S_i^t A_i^t, & \text{if } A_i^t \geq 0, S_i^t \geq 0 \\ A_i^t + S_i^t(1 + A_i^t) = A_i^t + S_i^t + S_i^t A_i^t, & \text{if } A_i^t < 0, S_i^t < 0, |A_i^t|, |S_i^t| \leq 1 \\ (A_i^t + S_i^t) / (1 - \min(|A_i^t|, |S_i^t|)), & \text{otherwise} \end{cases} \quad (3)$$

## 3. The Negotiation Environment

The environment used in this work endeavors to support the negotiation decision making process aiming at facilitating the negotiator's knowledge acquisition through a group of suggestive synthetic interfaces and reports, which were developed through several innovative technologies, as shown in Figure 1.



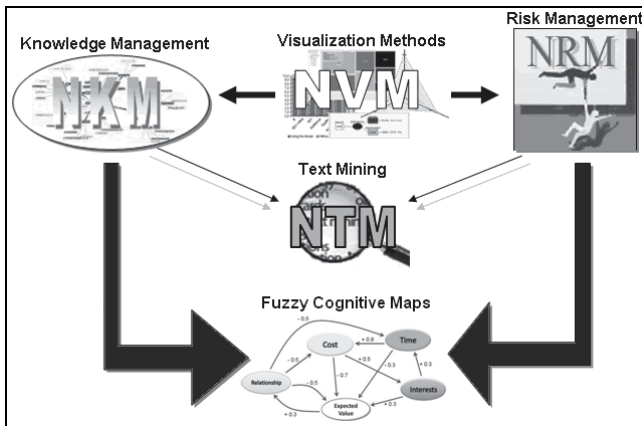


Figure 1: The Negotiation Support System

The Knowledge Management module concerns several preparation interfaces, which manage the negotiation knowledge through a tutorial guide. In the preparation step, the negotiator is invited to fill in some forms to keep the negotiation's data up-to-date. These forms work as a checklist, guiding the negotiation based on key elements, such as interests, options, power, concessions, context, relationship, criterion, cognition, compliance and time.

The Risk Management module attempts to evaluate the negotiation risks through risk management methodologies. There are several options to try to capture the negotiator's perception about the risk influence. For instance, interfaces to measure risks of negotiation's price and time provide qualitative and quantitative views, while relationship's and interest's interfaces allow only qualitative measurements. After risk identification step, it is possible to calculate the expected value of each risk and, then, a negotiation's weighted average is estimated. This estimation is used as a component of knowledge management in the negotiation; thus, the information stored in this module is another input to manage the whole negotiation knowledge.

Both management modules (Knowledge and Risk) use Text Mining techniques to show possibilities to share and reuse the knowledge collected. Besides, from the information stored in such modules, it is possible to enhance negotiator's knowledge acquisition through a group of suggestive synthetic reports, which were developed through Visualization Methods.

Based on the stored records and through visualization methods, experienced negotiators may design negotiation models to store this information and also to help future similar deals. In this context, FCM interfaces make available tools to develop negotiation maps as well as to view fuzzification graphics and statistics, as depicted in Figure 2.

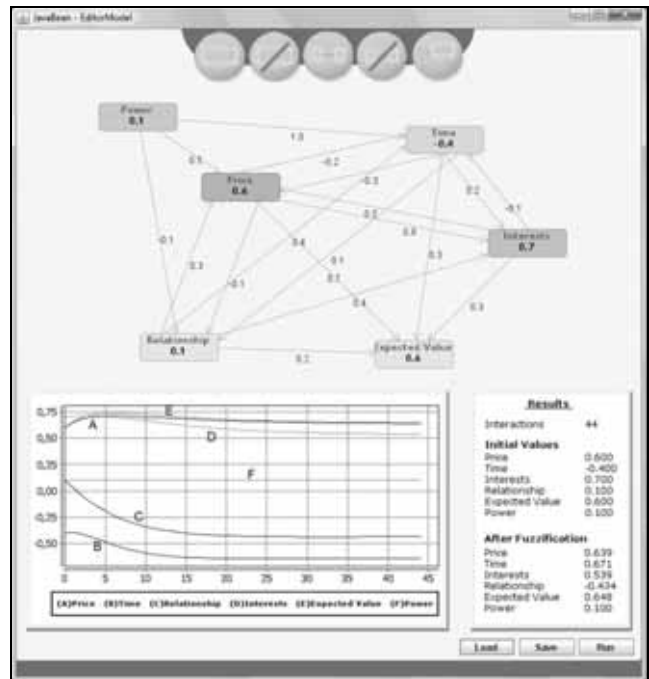


Figure 2: Interface to Create Negotiation Maps

Figure 2 shows an example of negotiation modeling and fuzzification tests to ensure that model converges as expected. From this tool, fuzzy analyses were conducted on real contract negotiations of software development, as explained in the following sections.

#### 4. Modeling the Experiment

From the use of historical negotiations – in this case, with the same client, this work obtained associations among the negotiation elements through real negotiations experiments. Figure 3 shows the connections among the employed elements.

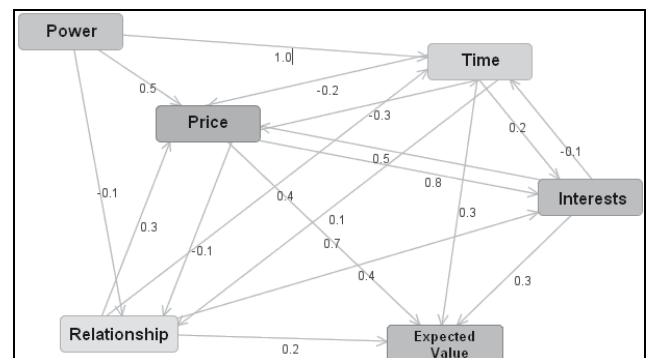


Figure 3: Negotiation Elements Map

In such experience, negotiations carried out during software contract negotiations were analyzed. The real names were changed to guarantee the confidentiality of the

parties involved. Copp is an IT group which researches and develops software. Copp employs around 150 professionals (managers, developers and research staff), including the authors of this article. In this case study, Copp was the Service Supplier. The Client of this negotiation was BPetrol institution, a global oil exploration and production company, operating especially in Brazil.

Software developments to a specific client and a variety of actors were the main criteria to choose this case study. What is more, all negotiations had at least 3 rounds. This parameter was set to avoid distortion in statistical results. As a comparison inputs, the case study presents six distinct negotiation elements: contract's price, wait time to reach an agreement, supplier's interests, counterparts' relationship, the negotiation's expected value and the power of influence, as better described in Table 1.

Table 1: Concept elements of the FCM proposed model

Element	Description
Price	<ul style="list-style-type: none"> <li>Represents the value of the proposal in the supplier point of view. In this case, the more expensive, the better.</li> <li>Price directly influences supplier's interests and expected value but can increase the negotiation period of time and also reduce the level of relationship among counterparts.</li> </ul>
Time	<ul style="list-style-type: none"> <li>Reflects how long the negotiation delays. For a supplier, the faster, the better means high values (positive) reflect fast negotiations while low values (negative) indicate slow negotiations.</li> </ul>
Interests	<ul style="list-style-type: none"> <li>Represents supplier's strategic concerns, i.e., the premeditated goals.</li> <li>Generally, several interests implies in high costs (and, consequently, high prices) and elevated expected value. On the other hand, supplier's interests may influence negatively the negotiation time.</li> </ul>
Relationship	<ul style="list-style-type: none"> <li>Level of good rapport and communication between counterparts. In general, this element influences in better prices and also increases the chances of raise interests, and, consequently, affects positively the negotiation expected value.</li> </ul>
Expected Value	<ul style="list-style-type: none"> <li>Represents how interesting the deal is from the supplier's perspective, considering key negotiation elements, strategic situations, valuable contexts and the agreement obligations.</li> <li>From the seller's point of view, high prices, several supported interests and good relationship increases the expected value while the delay to reach an agreement (time element) decreases the expected value.</li> </ul>

Power	<ul style="list-style-type: none"> <li>Represents power aspects which can influence the negotiation, such as power of authority, power of information, expert power or even personal persuasive power.</li> <li>This element influences time, price and relationship. Besides, the power implication is constant once it does not change through other variables, it means, its impact value does not decrease during the fuzzification.</li> </ul>
-------	---

Experienced negotiators, supported through historical values of client's negotiations, defined the association between the concept elements as well as the degree of influence among these factors, as illustrated in Table 2. The underlying weights and the values of the activation levels of the participating concepts are illustrated in a six-scale scheme showed in Table 3.

Table 2: Influences between concepts in the FCM negotiation model expressed as numerical weight values (column is the source)

	Pr	Ti	In	Re	EV	Po
Price		-0,3	0,8	-0,1	0,7	0
Time	-0,2		0,2	0,1	0,3	0
Interests	0,5	-0,1		0	0,3	0
Relationship	0,3	0,4	0,4		0,2	0
Expected Value	0	0	0	0		0
Power	0,5	1	0	-0,1	0	

Table 3: Linguistic terms and their respective values

very bad	bad	regular	good	excellent
-1 to -0.5	-0.51 to 0	0.01 to 0.4	0.41 to 0.7	0.71 to 1

## 5. Experimental Results

This section presents three scenarios which were used to investigate the efficacy of the model illustrated in Figure 3. The first and third scenarios represent the extreme cases of the worst and best circumstances in terms of parameter values that hinder or promote successful conclusion of the negotiation. The second case lies somewhere in between. Each negotiation case is characterized by initial activation levels for the participating concepts that reflect, to a high extent, what the case stands for, i.e. in favor or against the deal. The values of the fuzzy range are denoted by the linguistic value as follows:

### Negotiation 1: The worst agreement

- Price → bad (-0.4)
- Time → regular (0.1)
- Interests → regular (0.2)
- Relationship → bad (-0.4)
- Expected Value → bad (-0.4)
- Power → bad (-0.1)

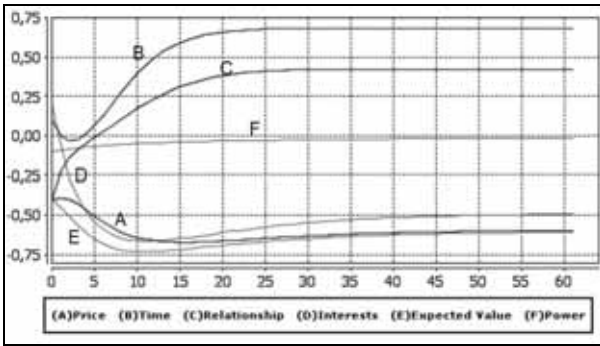


Figure 4: Stabilization graph of the Negotiation 1 in equilibrium after 65 iterations

**Negotiation 2: A medium agreement**

- Price → good (0.6)
- Time → bad (-0.4)
- Interests → good (0.7)
- Relationship → regular (0.1)
- Expected Value → good (0.6)
- Power → regular (0.1)

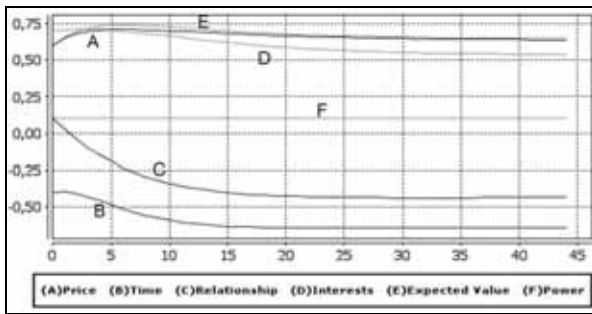


Figure 5: Stabilization graph of the Negotiation 2 in equilibrium after 44 iterations

**Negotiation 3: The best agreement**

- Price → excellent (0.8)
- Time → good (0.7)
- Interests → good (0.7)
- Relationship → regular (0.2)
- Expected Value → excellent (0.8)
- Power → excellent (0.8)

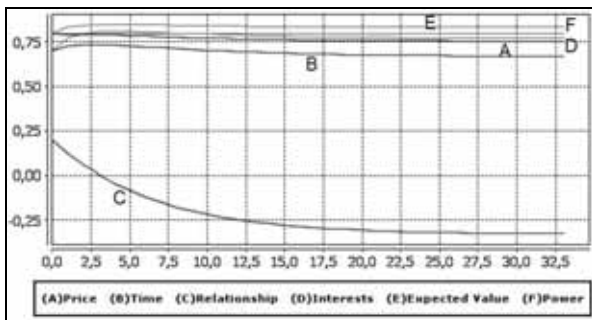


Figure 6: Stabilization graph of the Negotiation 3 in equilibrium after 34 iterations

Each involved case study executed the enough number of map iterations until it reaches in a final immutable situation, characterized by equilibrium. For each respective iteration, the new activation level value for each concept was calculated using equations (1) to (4), as previously explained. The final values of the activation levels are listed in Table 4, while Figures 4, 5 and 6 present graphical representation of each negotiation equilibrium state.

Table 4: Final activation levels of the concepts in the FCM negotiation model

	Negot. 1	Negot. 2	Negot. 3
<b>Price</b>	-0.596	0.639	0.755
<b>Time</b>	0.679	-0.640	0.671
<b>Interests</b>	-0.493	0.539	0.772
<b>Relationship</b>	0.418	-0.434	-0.326
<b>Expected Value</b>	-0.608	0.648	0.833
<b>Power</b>	-0.100	0.100	0.800

Analyzing the results of Table 4 and the Figures 4, 5 and 6, it is possible to notice that the model behaved as expected. More specifically, in the worst and best scenario cases the expected value of the negotiation concept stabilizes at -0.608 and 0.833, which suggests that the final outcome will eventually be negative and positive, respectively. The rest of the concepts also behaved as expected.

In Negotiation 1, both price and expected value are driven to even more negative values than originally started, while it is interesting to note that Interest becomes negative, which indicates that senior management stops participating in "lost" cases and devotes their time to other more beneficiary projects. Additionally, Relationship becomes positive indicating that trust and good communication may not be hampered in cases in which the negotiation ended without a consensus due to infeasible development that result from unsatisfactory time and price projections.

In Negotiation 2, it is possible to verify that almost all values keep established, mainly, Price and Expected Value. To maintain these important aspects, the Relationship was negatively affected. Besides, Figure 5 shows a little decrease in the Interests and also in the Time to get the agreement, which means negotiators were at the negotiation table longer than expected.

Reversed result is observed for the Negotiation 3 (best case) which justifies the correctness of the model in capturing properly the dynamics behind such promising negotiation scenery. In such case, an external aspect – the supplier’s Power of influence – was determinant to

maintain established price, time, interests and, consequently, the expected value. It is possible to notice a decrease in relationship level, which is comprehensive once all other factors push this level down.

## 6. Conclusions

This work aimed at addressing a strategy to facilitate the negotiation preparation through models applied to fuzzy cognitive maps approach. In the course of this paper, key negotiation elements were identified as well as the correlation weight among them. From this association, fuzzification inferences were applied to verify models convergence and, consequently, the reliability of the involved negotiation expected values.

The work also examined the importance of evaluating this approach through real negotiation experiments. Three hypothetical scenarios were executed taking into consideration key negotiation concepts. The results showed that the method is promising as the model reacts as expected.

Furthermore, fuzzification simulations are advantageous once they may be very dynamic, flexible and pragmatist, capturing the scenery, especially considering these main elements in classic negotiations. Using an intuitive interface, the negotiator may improve the coefficient of certainty in relation to the impact of each element change, and then, prioritize the preponderant ones. The regular use of this approach can make negotiations more objective due to better-defined criteria.

Conclusively, for future work, the innovative tool proposed may be further examined to involve other supplementary elements to the software, which may also be included in the assessment model of Certainty Neuron Fuzzy Cognitive Maps (CNFCM), and make inferences in different negotiation areas to examine the methods generalization to other backgrounds. Moreover, the expectation is that the tool can automatically suggest adjustments in likelihood and impact parameters as well as present a list of successful reactions.

## 7. References

[1] PMBOK. "Project Management Body of Knowledge", 2004 Edition. Project Management Institute, <http://www.pmi.org>, 2004.

[2] Rodrigues, S. A. et al. "A Case Study for a Complex Negotiation Analysis on Software Development Projects". In: GDN 2008 Group Decision and Negotiation Meeting, Coimbra, 2008.

[3] Miller, G. "The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information". *Psychological Review*, Vol.63, No. 2, 1956.

[4] Kosko, B., "Fuzzy Thinking. The New Science of Fuzzy Logic", London: Harper Collins, 1994.

[5] Papatheocharous, F. et al. "Qualitative Software Cost Estimation Using Fuzzy Cognitive Maps", AISEW - Artificial Intelligence Techniques in Software Engineering Workshop at 18th European Conference on Artificial Intelligence, Patras, Greece, 2008.

[6] Fisher, R. et al, "Das Harvard Konzept", 2nd edition. Campus Verlag, Frankfurt, 2002.

[7] Duzert, Y. org. "Manual de negociações complexas". Rio de Janeiro: Editora FGV, 2007.

[8] Susskind, L., and Cruikshank, J., "Breaking the Impasse": Consensual Approaches to Resolving Public Disputes. Basic Book, New York, USA, 1987.

[9] Fisher, R., Ury, W. "Getting To Yes": Negotiating an Agreement Without Giving In. Boston, Century Business, 1991.

[10] Harvard. Negotiation, "The Harvard Business Essentials Series" (Paperback), USA: Harvard Business School Publishing Corporation, 2003.

[11] Raiffa, H. et al. "Negotiation Analysis: the Science and Art of Collaborative Decision Making". Harvard University Press, Cambridge, England, 2002.

[12] Bazerman, M. H., "Judgment in Managerial Decision Making". 5th ed. New York: Wiley, 2002.

[13] Rawls, J. "A theory of justice. Harvard" University Press, Cambridge, England, 1971.

[14] Adams, C. R. and Hicks, R. D. "Preparation For Trial". The Harrison Company, 300 p, 2001.

[15] Tardy, T. "The Brahimi Report: Four Years On". Proceedings of a Workshop held at the GCSP, 2004.

[16] Kennedy, G. "Essential Negotiation", The Economist Newspaper Ltd., 240 p, 2004.

[17] Lewicki, R. et al. "Negotiation". McGraw-Hill, Boston, 3rd edition. 202 p, 1999.

[18] Tsadiras, A.K. and Margaritis, K.G. "Cognitive Mapping and the Certainty Neuron Fuzzy Cognitive Maps", *Information Sciences*, Vol. 101, 109-130, 1997.

[19] Tsadiras, A.K. and Margaritis, K.G. "The MYCIN Certainly Factor Handling Function as Uninorm Operator and its Use as Threshold Function in Artificial Neurons", *Fuzzy Set and Systems*, Vol. 93, 263-274, 1998.

# Value-Based Software Quality Modeling

Naeem Seliya

Computer and Information Science  
University of Michigan – Dearborn  
4901 Evergreen Rd., Dearborn, MI 48128  
Email: nseliya@umich.edu

Taghi M. Khoshgoftaar

Computer Science and Engineering  
Florida Atlantic University  
777 Glades Rd., Boca Raton, FL 33431  
Email: taghi@cse.fau.edu

**Abstract**—This study is unique in incorporating cost-sensitive learning techniques during the model-training process of building software quality estimation models, such as predicting program modules as fault-prone or not-fault-prone. Such models are usually built from knowledge of past projects and then evaluated using prediction error rates or performance metrics derived from the four basic metrics: true positive rate, false positive rate, true negative rate, and false negative rate. To date, most studies have evaluated the cost aspects of software quality models using the expected cost of misclassification after model training. In this study we investigate the strategy of using a cost-sensitive learning technique, MetaCost, during the training process of C4.5 and Naive Bayes classification models. Software practitioners can use this approach to obtain a direct insight into the cost-based performance of the trained software quality model. A large case study of four software measurement datasets, two classification algorithms, one cost-sensitive learning technique, and a wide range of cost ratio values reveals the empirically-validated benefits of using cost-sensitive learning as a useful technique during defect prediction modeling.

Keywords: fault prediction; cost-sensitive learning; software measurements; classification models.

## I. INTRODUCTION

Software quality prediction models have been extensively investigated toward improving the quality and reliability of software-based systems [5], [7], [11], [13]. Software measurement and defect data from past projects or releases are used as base knowledge to build prediction models that estimate the software quality of the under-development project. This allows the testing and inspection team to focus their efforts on low-quality areas thus maximizing the benefits gained from resources expended on additional testing and inspections.

Building software quality models involves utilizing the knowledge base available from the data mining and machine learning communities. For example, binary classification models are often used for predicting the quality of program modules as either fault-prone (*fp*) or not-fault-prone (*nfp*). One can find various techniques that have been empirically validated for building useful software prediction models, such as decision trees, logistic regression, case-based reasoning, etc [5], [7], [11]. Once such models are built, their predictive performance is evaluated with respect to various performance metrics, such as misclassification error rates, overall accuracy, F-Measure, Recall, Precision, etc.

A two-group (positive or fault-prone and negative or not-fault-prone) classification problem has a confusion matrix consisting of four cells, i.e., true positive, true negative, false positive, and false negative. If the positive class represents *fp* modules and the negative class represents *nfp* modules, then a false positive indicates an error in which an *nfp* program module is incorrectly classified as *fp*. A false negative indicates an error in which an *fp* program module is incorrectly classified as *nfp*. A false negative is the more serious error type as it represents a lost opportunity to detect an actual *fp* module. Whereas a false positive leads to wasted resources due to inspection of an already good quality program module.

Clearly, the cost of misclassifying an *fp* program module is different than the cost of misclassifying an *nfp* program module. This brings about the natural question of how do we incorporate this issue during the task of building and evaluating software quality prediction models. Analysts have attempted to use the total cost of misclassification (as explained in the next section) as a performance metric that would provide insight into the penalties (of misclassifications) associated with a given software quality model. However, such strategies are considered only after the model training has been completed.

In this study, we investigate using cost-sensitive learning techniques during the actual process of training a software quality model. To our knowledge, this is the first study to combine cost-sensitive learning techniques with a given classification algorithm for training software quality prediction models. Compared to accuracy measurements, such as error rates, ROC curve, F-Measure, etc., a software practitioner is more interested in the cost associated with a given software quality estimation model. This study provides a solution to the practitioner in terms of building software quality models at a specific misclassification cost.

We take an empirical approach to presenting and validating our work. The large-scale case study involves software measurement data obtained from four high-assurance software systems; two independent classification algorithms, namely C4.5 decision tree and Naive Bayes [14]; and a cost-sensitive learning technique, namely MetaCost [2]. While several other classifiers and cost-sensitive learning techniques were investigated, those results cannot be presented due to paper-size limitations.

The remainder of this paper is structured as follows: Section II discusses some of the most relevant related works in the context of this study; Section III summarizes the MetaCost cost-sensitive learning technique, the performance metrics used to evaluate the different software quality models, and the two classification algorithms used; Section IV details the software measurement datasets, experimental settings, and the results obtained; and Section V concludes this paper with a summary of our empirical investigation and provides suggestions for future work.

## II. RELATED WORK

In this section we limit our discussion to some of the related literature that focuses on incorporating misclassification costs during software quality modeling. In the case of classification models, the expected cost of misclassification (ECM and its variations – see Equation 2 in Section III-C) is commonly used to evaluate the cost-based performance of a given classifier [8], [11], [12]. In some cases, the total cost of misclassification is normalized with respect to the number of instances (program modules) in the dataset.

In the context of software quality modeling, we have previously investigated ECM for evaluating software quality models [11], [12].

While a logical approach to evaluating the cost aspects of a fault prediction model, ECM is not incorporated during the model-training process – this strategy is followed by most existing related literature. A software quality model’s misclassification cost is only computed after it is trained (i.e., during the model evaluation process) and applied to a target dataset. This study is unique in incorporating cost-sensitive learning during the training process of building a software quality model.

We have also used evolutionary techniques to train models that are optimized for multiple objectives, including expected cost of misclassification [10]. The black-box and non-traditional nature of evolutionary techniques tend to limit their appeal to software quality practitioners. In addition, with ECM as the performance evaluation metric, one would have to evolve software quality models for different misclassification costs. The problem of relatively slow training times and the tedious task of optimizing evolutionary parameters makes genetic programming and genetic algorithms less attractive to analysts.

In a relatively recent study, Drummond and Holte introduced cost curves as a visual representation of the performance of binary classifiers across all class distributions and misclassification costs [4]. It is stated that ROC (receiver operating characteristic) curves and cost curves are very closely related, i.e. there is a bidirectional point/line duality between them. This implies that a point in ROC space is represented by a line in cost space and a line in ROC space is represented by a point in cost space, and vice versa. We note that cost curves are determined only after the model-training process of building a binary classifier is completed.

Jiang et al. [8] apply the cost curves proposed by Drummond and Holte [4] in the context of fault prediction modeling. Based on a study of multiple software measurement datasets, they recommend adopting cost curves as one of the standard methods for evaluating fault prediction models. However, as stated previously, cost curves are not incorporated into the actual training process of building a classifier. More specifically, cost curves are used as a performance metric during model evaluation. In contrast to using cost curves for performance evaluation, we investigate using cost-sensitive learning techniques during the model-training process of building a software quality model. The software quality analyst is more interested in what factors are considered during the model-training process. Since cost curves are very closely related to ROC curves [4], they would tend to suffer from the same limitations that ROC curves suffer in the context of its practical appeal and usage to the software quality analyst.

### III. MODELING METHODOLOGY

#### A. Classification Algorithms

The two classification algorithms used in our study to build software quality models are C4.5 decision tree and Naive Bayes. We use the WEKA [16] data mining tool to conduct our empirical studies with these two learners. These two classification algorithms were selected based on their common use both in the software engineering and machine learning communities.

C4.5 is the benchmark decision tree learning algorithm proposed by Quinlan [14]. C4.5 is among the most commonly used learning algorithms in data mining research. The decision tree is built using an entropy-based splitting criterion stemming from information theory. C4.5 improves Quinlan’s older ID3 decision tree algorithm by adding support for tree pruning and dealing with missing values and numeric attributes (software metrics). The WEKA version of C4.5 is called J48 [16], and we use its default parameters for C4.5 in our experiments.

Naive Bayes [16] (NB) is a quick and simple classifier that utilizes the Bayes rule of conditional probability. It is “naive” in that it assumes that all predictor variables are independent. Although this assumption rarely holds true in real-world data, Naive Bayes has been shown to often perform well even in the presence of strong attribute dependencies [3]. We use the default parameters for Naive Bayes in our experiments.

In the context of this study, we have conducted similar empirical studies with several other classification algorithms. However, those results are not presented due to relative similarity of conclusions.

#### B. Cost-Sensitive Learning with MetaCost

Various techniques exist for incorporating cost-based learning during classification modeling, and MetaCost is one such technique. Proposed by Domingos [2] for making any error based classifier cost sensitive, MetaCost is based on *Bayes optimal prediction*. More specifically, if for a given example (instance)  $x$ , we know the probability of each class  $j$ , i.e.,  $P(j|x)$ , the Bayes optimal prediction for  $x$  is the class  $i$  that minimizes the *conditional risk*,  $R(i|x)$  (see Equation 1), which is the expected cost of predicting that  $x$  belongs to class  $i$ . The Bayes optimal prediction is certain to achieve the lowest possible overall cost over all possible examples  $x$ , weighted by their probabilities  $P(x)$ .  $C(i, j)$  and  $P(j|x)$  together with Equation 1 imply a partition of the instance space  $X$  into  $j$  regions, such that class  $j$  is the optimal (i.e., lowest cost) prediction in region  $j$  [2].

$$R(i|x) = \sum_j P(j|x)C(i, j) \quad (1)$$

MetaCost modifies the labels of training data instances so that their labels represents their “optimal classes.” This is achieved by learning multiple classifiers, and using the result of each classifier as a vote in determining the probability that an instance belongs to a specific class. Bagging [1] is used to build an ensemble of learners. Samples (program modules) are taken, with replacement, from the training dataset, creating a new training dataset of the same size. This is repeated  $m$  times (we use  $m = 10$ ) with  $m$  models being trained using the resampled datasets. The probability that an instance (program module) belongs to a class is based on the fraction of votes it received, or an unweighted average of the probability estimates of the  $m$  models. Based on the estimated probability and the cost ratio, new class labels are assigned. The newly labeled training dataset is then used by the given classification algorithm to produce a cost-sensitive software quality prediction model. The algorithm for MetaCost proposed by Domingos [2] is provided in Table I.

#### C. Performance Metrics

In this study, the performance of the two classifiers is evaluated primarily by comparing their *per-example-cost* (PEC), which is the total cost of misclassification divided by the number of instances in the dataset. The total cost of misclassification (TC) is given by,

$$TC = \#fpos \times C(1, 0) + \#fneg \times C(0, 1) \quad (2)$$

where,  $\#fpos$  is the number of  $nfp$  modules predicted as  $fp$ ,  $\#fneg$  is the number of  $fp$  modules predicted as  $nfp$ ,  $C(1, 0)$  is the cost of classifying an  $nfp$  module as  $fp$ , and  $C(0, 1)$  is the cost of classifying an  $fp$  module as  $nfp$ . Since the exact costs of misclassifications are unknown during modeling and analysis, different values for the cost ratio, i.e.  $\frac{C(0,1)}{C(1,0)}$ , are used depending on the characteristics of the software system.

In addition to the cost aspects of the various software quality models, we also present their *F-Measure* (F-Meas) values. This

TABLE I  
METACOST ALGORITHM

---

Inputs:

- $S$  is the training dataset
- $L$  is a classification algorithm
- $C$  is a cost matrix
- $m$  is the number of resamples to generate
- $n$  is the number of examples in each resample
- $p$  is *True* iff  $L$  produces class probabilities
- $q$  is *True* iff all resamples are to be used for each example

Procedure MetaCost ( $S, L, C, m, n, p, q$ )

For  $i = 1$  to  $m$

- Let  $S_i$  be a resample of  $S$  with  $n$  examples
- Let  $M_i =$  Model produced by applying  $L$  to  $S_i$

For each example  $x$  in  $S$

For each class  $j$

Let  $P(j|x) = \frac{1}{\sum_{i=1}^m} \sum_i P(j|x, M_i)$

Where

- If  $p$  then  $P(j|x, M_i)$  is produced by  $M_i$
- Else  $P(j|x, M_i) = 1$  for the class predicted by  $M_i$  for  $x$ , and 0 for all others
- If  $q$  then  $i$  ranges over all  $M_i$
- Else  $i$  ranges over all  $M_i$  such that  $x \ni S_i$

Let  $x$ 's class =  $\text{argmin}_i \sum_j P(j|x)C(i, j)$

Let  $M =$  Model produced by applying  $L$  to  $S$

Return  $M$

---

performance measurement is based on two information retrieval metrics, Recall (or effectiveness) and Precision (or efficiency), where Recall is the true positive rate and Precision is the ratio of the number of true positives to the sum of the number of true positives and the number of false positives. When Recall and Precision are given equal importance, the F-measure is computed as  $\frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$ .

#### IV. EMPIRICAL CASE STUDY

##### A. Software Measurement Data

The four software measurement datasets used in our case study are high-assurance systems from different application domains. These datasets were selected based on different values for dataset size and the relative proportion of *fp* modules. Typically, a function, subroutine, or method is considered as a program module for these systems [9], [11], [15]. The use of specific software metrics was governed primarily based on their availability for analysis purposes.

The CCCS-8 project is a large military command, control, and communication system implemented in Ada. The dataset consists of 282 modules, of which 27 (9.57%) are *fp* and 255 are *nfp*. A program module in CCCS-8 is characterized by eight software product metrics, including Halstead's and McCabe's complexity metrics [6]. The reader is referred to a prior work, Khoshgoftaar and Allen [9], for additional details on the software metrics used for this system.

The KC2 project, written in C++, is the science data processing unit of a storage management system used for receiving and processing ground data for missions. This data (and PC1) is available through the NASA Metrics Data Program. The 13 software product metrics characterize 520 program modules, of which 106 (20.38%) are *fp* and 414 are *nfp* [15]. The software metrics include Halstead's and

MCCabe's complexity metrics and statement (i.e. lines of code) metrics.

The PC1 project is flight software for an earth orbiting satellite that is no longer operational. The software measurement dataset contains 1107 modules characterized by 13 product metrics, the same as those for KC2. There are respectively 76 (6.87%) *fp* and 1031 *nfp* program modules [15].

The SP4 project data contains software metrics and defect data for a release of a large-scale telecommunications system written in Protel [11]. The 42 software metrics characterize different attributes of an SP4 program module, and include both product and process metrics. There are a total of 3978 modules in SP4, of which 92 (2.31%) are *fp* and 3886 are *nfp*. The reader is referred to a prior work, Khoshgoftaar and Seliya [11], for additional details on the software metrics used for this system.

##### B. Experimental Settings

The C4.5 and Naive Bayes learners were built with (MetaCost), and without (referred to as None), cost-sensitive learning for each of the four software measurement datasets. The two classifiers were used with their default settings in WEKA [16].

When MetaCost is used during software quality modeling, the cost ratios used include 10, 15, 20, 25, 30, 40, and 50. Generally speaking, for high-assurance systems such as those of our case study, a cost ratio of 25 is considered appropriate. In order to provide the analyst with a better insight into cost-sensitive software quality modeling, we consider a wide range of cost ratio values. One may consider a different set of cost ratio values depending on their appropriateness to the software project under consideration.

In our study, a classifier is built using 10 fold cross validation, and this process is repeated 10 times (i.e. 10 runs) to avoid any bias due to a lucky/unlucky split when obtaining the 10 folds for cross validation. The empirical results shown in the next section represent the classifier performance averages across the 10 runs. Thus, with four datasets, two learners, two cost-sensitive learning techniques (MetaCost and None), five cost ratios, 10 cross validation folds, and 10 runs, a total of 8000 models/cost-ratio combinations were constructed and evaluated.

##### C. Results and Analysis

The modeling results for the two learners when built without any cost-sensitive learning are summarized in Table II. The results obtained after building the classifiers in conjunction with MetaCost at different cost ratios are summarized in Table III for C4.5 and Table IV for Naive Bayes (NB). These tables show the four types of error rates and the F-Measure value.

According to Table II, the Naive Bayes learners generally provided better or similar F-Measure values than the C4.5 learners. When C4.5 is applied with MetaCost, the F-Measure values tend to decrease with an increase in the cost ratio. However, when Naive Bayes is applied with MetaCost, the F-Measure values tend to either remain steady or decrease slightly with an increase in the cost ratio. For both learners, an increase in the cost ratio brings about (as expected) an increase in both the true positive rate and the false positive rate. While we present the F-Measure values for completeness sake, our focus in this paper is on comparing the different learners based on their cost of misclassifications.

Since we emphasize a cost-sensitive learning approach to train a software quality model, the total cost of misclassification values for the two learners at different cost ratios are shown in Tables V and VI. The tables show the TC values for the two learners built

without MetaCost and with MetaCost. In the case of C4.5, learning with MetaCost clearly lowers (for all cost ratios and all datasets) the TC values as compared to learning without MetaCost. This is also generally true for NB, except for PC1 and SP4 at the respective cost ratios of 40 and 50. For these four cases, NB without MetaCost provides slightly better (lower) TC values. This may be indicative of the fact that C4.5 benefits more with MetaCost than NB, an observation that is discussed in the remainder of this section.

The per-example-cost, or PEC, of the software quality models built with, and without, cost-sensitive learning are plotted against the different cost ratios in Figures 1 through 4. In the case of CCCS-8, the smallest of the four datasets investigated, there is a clear advantage in using MetaCost during learning over None (i.e., not using any cost-sensitive learning). The improvement in PEC (from no-cost-sensitive learning to cost-sensitive learning) increases with the increase in the cost ratio. When comparing how MetaCost aids C4.5 and NB, adding MetaCost to NB provides a relatively lower improvement in PEC (over NB without MetaCost) as compared to C4.5 with MetaCost. Among the four learners, NB with MetaCost provides the lowest PEC values in absolute terms, followed by C4.5 with MetaCost, NB without MetaCost, and C4.5 without MetaCost.

For the second smallest of the four datasets, i.e. KC2, the C4.5 and NB learners without MetaCost demonstrate relatively similar performance across most of the cost ratios, with C4.5 edging out NB at high cost ratios. Among the four learners, C4.5 with MetaCost is clearly by-far the best model, followed by NB with MetaCost, C4.5 without MetaCost, and NB without MetaCost – the latter two provide relatively similar performances. An interesting observation here is that adding MetaCost to C4.5 provides a more dramatic improvement (over C4.5 without MetaCost) as compared to adding MetaCost to NB (over NB without MetaCost). This observation was also noted with the CCCS-8 dataset. Once again, for both learners the reduction (from no-cost-sensitive learning to cost-sensitive learning) in PEC values is greater for larger cost ratios.

In the case of PC1, NB barely shows any improvement with MetaCost over the different cost ratios. In contrast, C4.5 with MetaCost once again shows a dramatic reduction in the PEC values over C4.5 without MetaCost. Among the four models, C4.5 with MetaCost is once again (also for KC2) clearly the best model, followed by NB with MetaCost, NB without MetaCost, and C4.5 without MetaCost. The latter three models provide relatively similar performances.

For the largest of the four datasets, i.e. SP4, the improvement obtained from using MetaCost with C4.5 is not as dramatic as the CCCS-8, KC2, and PC1 datasets. However, that improvement is still slightly better than when NB is used with MetaCost. In absolute terms, among the four models the best model is NB with MetaCost, very closely followed by NB without MetaCost, C4.5 with MetaCost, and C4.5 without MetaCost. Among the four datasets, this is the only time when C4.5 with MetaCost is not better than NB without MetaCost. The unique observations made for SP4 is likely due to its extremely low proportion of *fp* modules (2.31%) compared to the other datasets.

One of the general conclusion from our study is that while Naive Bayes generally provided better or similar F-Measure values than C4.5, the latter benefits much more from adding cost-sensitive learning during the training process of a software quality estimation model. In addition, while various characteristics of software measurement datasets can influence the outcome of prediction models, we have observed that the improvement gained with cost-sensitive models is affected by the relative proportion of *fp* modules in the training data. More specifically, in our study, the improvement from no-cost-

TABLE II  
RESULTS WITHOUT METACOST

Data	TPR	TNR	FPR	FNR	F-Measure
C4.5 Learner					
CCCS-8	0.5617	0.9624	0.0376	0.4383	0.5533
KC2	0.4491	0.9163	0.0837	0.5509	0.4893
PC1	0.2375	0.9887	0.0113	0.7625	0.3273
SP4	0.0888	0.9932	0.0068	0.9112	0.1221
Naive Bayes Learner					
CCCS-8	0.7350	0.9553	0.0447	0.2650	0.6795
KC2	0.4220	0.9421	0.0579	0.5780	0.5008
PC1	0.3011	0.9327	0.0673	0.6989	0.2650
SP4	0.4744	0.8811	0.1189	0.5256	0.1465

TABLE III  
C4.5 RESULTS WITH METACOST

Cost Ratio	TPR	TNR	FPR	FNR	F-Measure
CCCS-8 Dataset					
10	0.7917	0.9239	0.0761	0.2083	0.6386
15	0.7817	0.9208	0.0792	0.2183	0.6174
20	0.7800	0.9161	0.0839	0.2200	0.6057
25	0.7900	0.9161	0.0839	0.2100	0.6112
30	0.7967	0.9078	0.0922	0.2033	0.5988
40	0.8217	0.8914	0.1086	0.1783	0.5815
50	0.8450	0.8722	0.1278	0.1550	0.5661
KC2 Dataset					
10	0.7883	0.7618	0.2382	0.2117	0.5801
15	0.8238	0.7391	0.2609	0.1762	0.5808
20	0.8630	0.5797	0.4203	0.1370	0.5190
25	0.9183	0.2339	0.7661	0.0817	0.3924
30	0.9727	0.0646	0.9354	0.0273	0.3502
40	0.9964	0.0090	0.9910	0.0036	0.3402
50	1.0000	0.0000	1.0000	0.0000	0.3386
PC1 Dataset					
10	0.5188	0.9087	0.0913	0.4812	0.3779
15	0.5927	0.8836	0.1164	0.4073	0.3764
20	0.6975	0.8285	0.1715	0.3025	0.3487
25	0.7600	0.7698	0.2302	0.2400	0.3138
30	0.8150	0.7339	0.2661	0.1850	0.3023
40	0.8575	0.6779	0.3221	0.1425	0.2829
50	0.8941	0.6230	0.3770	0.1059	0.2689
SP4 Dataset					
10	0.2490	0.9738	0.0262	0.7510	0.2074
15	0.2604	0.9726	0.0274	0.7396	0.2146
20	0.2601	0.9724	0.0276	0.7399	0.2110
25	0.2592	0.9722	0.0278	0.7408	0.2109
30	0.2668	0.9694	0.0306	0.7332	0.2073
40	0.2933	0.9620	0.0380	0.7067	0.2027
50	0.3297	0.9499	0.0501	0.6703	0.1919

sensitive learning to cost-sensitive learning tended to decrease with a decrease in the proportion of *fp* modules in the dataset. To a software practitioner the value of this study is the much-needed emphasis on cost-sensitive learning during the model building process of a software quality estimation model.

## V. CONCLUSION

This study presented a practical insight into the use of cost-sensitive learning techniques during the process of building and evaluating software quality prediction models. Instead of evaluating such models based on error rates and other related metrics,



TABLE IV  
NAIVE BAYES RESULTS WITH METACOST

Cost Ratio	TPR	TNR	FPR	FNR	F-Measure
CCCS-8 Dataset					
10	0.8817	0.8836	0.1164	0.1183	0.5970
15	0.8817	0.8758	0.1242	0.1183	0.5857
20	0.8817	0.8730	0.1270	0.1183	0.5810
25	0.8817	0.8718	0.1282	0.1183	0.5792
30	0.8817	0.8699	0.1301	0.1183	0.5761
40	0.8850	0.8683	0.1317	0.1150	0.5744
50	0.8900	0.8655	0.1345	0.1100	0.5717
KC2 Dataset					
10	0.5779	0.9000	0.1000	0.4221	0.5833
15	0.5919	0.8979	0.1021	0.4081	0.5908
20	0.5985	0.8969	0.1031	0.4015	0.5946
25	0.6014	0.8954	0.1046	0.3986	0.5950
30	0.6071	0.8950	0.1050	0.3929	0.5985
40	0.6089	0.8935	0.1065	0.3911	0.5981
50	0.6108	0.8928	0.1072	0.3892	0.5985
PC1 Dataset					
10	0.3670	0.8641	0.1359	0.6330	0.2279
15	0.3720	0.8608	0.1392	0.6280	0.2276
20	0.3761	0.8588	0.1412	0.6239	0.2276
25	0.3761	0.8558	0.1442	0.6239	0.2250
30	0.3761	0.8546	0.1454	0.6239	0.2241
40	0.3827	0.8518	0.1482	0.6173	0.2244
50	0.3839	0.8509	0.1491	0.6161	0.2243
SP4 Dataset					
10	0.6257	0.8136	0.1864	0.3743	0.1317
15	0.6312	0.8107	0.1893	0.3688	0.1310
20	0.6344	0.8084	0.1916	0.3656	0.1304
25	0.6377	0.8066	0.1934	0.3623	0.1300
30	0.6388	0.8057	0.1943	0.3612	0.1297
40	0.6431	0.8037	0.1963	0.3569	0.1293
50	0.6452	0.8020	0.1980	0.3548	0.1288

TABLE V  
TOTAL COST WITH C4.5

Cost Ratio	CCCS-8	KC1	PC1	SP4
No Cost-Sensitive Learning				
10	12.86	61.77	58.97	86.56
15	18.81	90.92	87.87	128.51
20	24.76	120.07	116.77	170.46
25	30.71	149.22	145.67	212.41
30	36.66	178.37	174.57	254.36
40	48.56	236.67	232.37	338.26
50	60.46	294.97	290.17	422.16
Cost-Sensitive Learning with MetaCost				
10	7.44	32.26	46.01	79.27
15	10.57	38.85	58.35	112.64
20	13.54	46.41	63.48	146.74
25	15.64	53.20	69.48	181.07
30	17.95	47.43	69.43	214.11
40	21.17	42.63	76.41	274.77
50	23.76	41.40	78.87	327.96

TABLE VI  
TOTAL COST WITH NAIVE BAYES

Cost Ratio	CCCS-8	KC1	PC1	SP4
No Cost-Sensitive Learning				
10	8.14	63.60	60.04	94.61
15	11.64	94.20	86.59	118.81
20	15.14	124.80	113.14	143.01
25	18.64	155.40	139.69	167.21
30	22.14	186.00	166.24	191.41
40	29.14	247.20	219.34	239.81
50	36.14	308.40	272.44	288.21
Cost-Sensitive Learning with MetaCost				
10	5.97	48.74	62.11	106.94
15	7.67	68.88	85.90	124.57
20	9.24	89.07	109.36	141.86
25	10.77	109.58	133.37	158.64
30	12.32	128.85	157.19	175.40
40	14.96	169.61	202.88	207.89
50	17.43	209.94	249.37	240.43

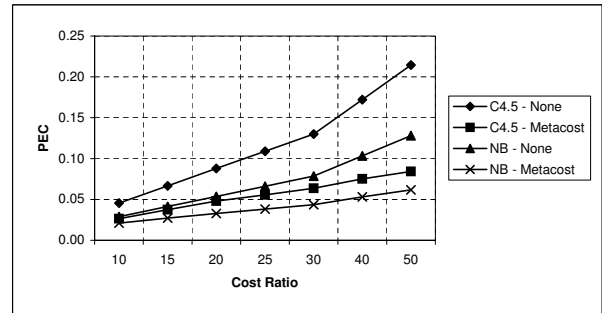


Fig. 1. CCCS-8 Dataset: None vs. MetaCost

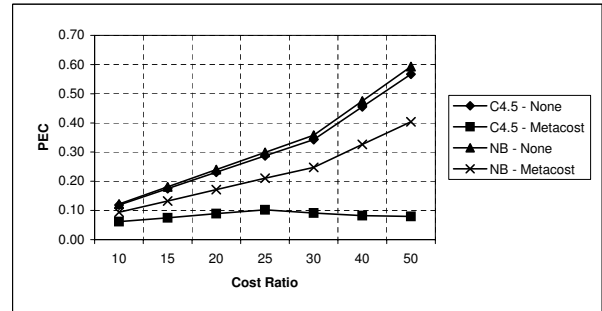


Fig. 2. KC2 Dataset: None vs. MetaCost

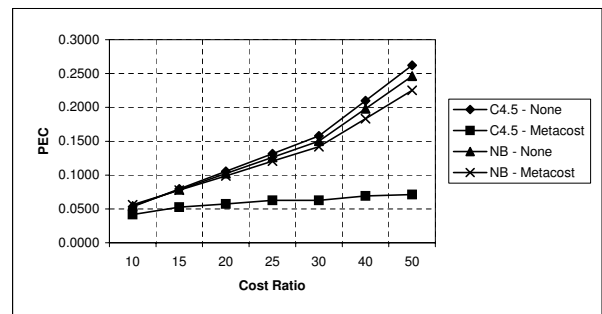


Fig. 3. PC1 Dataset: None vs. MetaCost

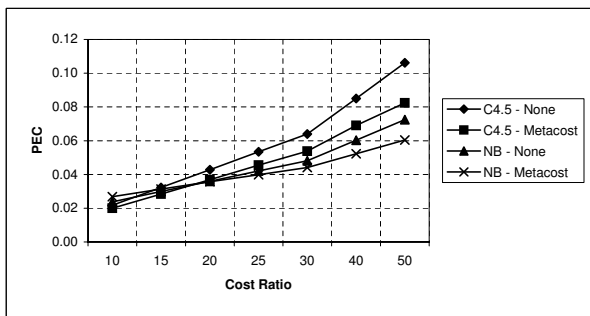


Fig. 4. SP4 Dataset: None vs. MetaCost

misclassification costs are considered during the training process, thus providing the analyst a better insight into which models are better suited at the preferred cost ratio. Associating a total cost of misclassification to the trained software quality estimation model, provides the software practitioner with a practical value that is more useful in software engineering project development.

The empirical case study involved building 8000 software quality estimation models reflecting the four software measurement datasets, two classification algorithms, two cost-sensitive learning strategies, and five cost ratio values. The case study dataset were all high-assurance software systems, and were of different sizes with each having a relatively different proportion of *fp* modules compared to the size of the dataset. The dataset sizes ranged from 282 program modules to 3978 program modules, while the proportion of *fp* modules ranged from 2.31% to 20.38% of the total number of modules in a given dataset.

The results from our experimentation clearly demonstrate the benefits of including a cost-sensitive learning technique during the model-training process. Within the scope of our case study, it was found that, compared to Naive Bayes, C4.5 benefited more greatly when incorporated with the MetaCost cost-sensitive learning technique. In addition, the relative proportion of *fp* modules in a given software measurement dataset seemed to affect the improvements gained from incorporating cost-sensitive learning during the model-training process.

Some directions for future work related to this study include: investigating other cost-sensitive learning techniques and comparing their performance with that of MetaCost; and providing further empirical validation to the conclusions of this paper by analyzing other software measurement datasets, possibly from other application domains and with different proportions of *fp* program modules in the dataset.

#### ACKNOWLEDGMENT

We thank the various members of the Empirical Software Engineering Laboratory and the Data Mining and Machine Learning Laboratory at Florida Atlantic University for their assistance with experimentation and manuscript reviews.

#### REFERENCES

- [1] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.
- [2] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of Knowledge Discovery and Data Mining*, pages 155–164, 1999.
- [3] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997.

- [4] C. Drummond and R. C. Holte. Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65(1):95–130, 2006.
- [5] K. E. Emam, S. Benlarbi, N. Goel, and S. N. Rai. Comparing case-based reasoning classifiers for predicting high-risk software components. *Journal of Systems and Software*, 55(3):301–320, 2001. Elsevier Science Publishing.
- [6] N. E. Fenton and S. L. Pfleeger. *Software metrics: A rigorous and practical approach*. 1997.
- [7] L. Guo, B. Cukic, and H. Singh. Predicting fault prone modules by the dempster-shafer belief networks. In *Proceedings of the 18th International Conference on Automated Software Engineering*, pages 249–252, Montreal, Quebec, Canada, October 2003. IEEE Computer Society.
- [8] Y. Jiang, B. Cukic, and T. Menzies. Cost curve evaluation of fault prediction models. In *Proceedings of the 19th International Symposium on Software Reliability Engineering*, pages 197–206, Seattle, WA, November 2008. IEEE Computer Society.
- [9] T. M. Khoshgoftaar and E. B. Allen. Logistic regression modeling of software quality. *International Journal of Reliability, Quality and Safety Engineering*, 6(4):303–317, December 1999.
- [10] T. M. Khoshgoftaar, Y. Liu, and N. Seliya. A multi-objective module-order model for software quality enhancement. *IEEE Transactions on Evolutionary Computation*, 8(6):593–608, December 2004.
- [11] T. M. Khoshgoftaar and N. Seliya. Comparative assessment of software quality classification techniques: An empirical case study. *Empirical Software Engineering Journal*, 9(3):229–257, 2004.
- [12] T. M. Khoshgoftaar, N. Seliya, and A. Herzberg. Resource-oriented software quality classification models. *Journal of Systems and Software*, 76(2):111–126, 2005.
- [13] M. C. Ohlsson and P. Runeson. Experience from replicating empirical studies on prediction models. In *Proceedings: 8th International Software Metrics Symposium*, pages 217–226, Ottawa, Ontario, Canada, June 2002. IEEE Computer Society.
- [14] J. R. Quinlan. *C4.5: Programs For Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.
- [15] N. Seliya and T. M. Khoshgoftaar. Software quality analysis of unlabeled program modules with semi-supervised clustering. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(2):201–211, March 2007.
- [16] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, California, 2nd edition, 2005.

# Predicting Maintainability expressed as Change Impact: a Machine-Learning-Based Approach

H. Lounis\*, M.K. Abdi\*\*, H. Sahraoui\*\*

\* Département d'Informatique, Université du Québec à Montréal  
Case postale 8888, succursale Centre-ville, Montréal QC H3C 3P8, Canada  
lounis.hakim@uqam.ca

\*\* Département d'Informatique et de Recherche Opérationnelle,  
Université de Montréal,  
CP 6128 succ Centre-Ville, Montréal QC H3C 3J7, Canada  
{abdimust, sahraouh}@iro.umontreal.ca

## Abstract

*This study has to be considered as another step towards the proposal of assessment/predictive models for software product quality. We consider in this work, both probabilistic and non-probabilistic machine-learning approaches, to predict maintainability, expressed as change impact, in object-oriented applications. Data obtained from a real system are exploited to empirically study causality hypotheses between some software internal attributes (coupling) and change impact. Models based on rules and decision trees present interesting performances; they also allow us the construction of a Bayesian network to analyze and predict change. Several scenarios are executed on the network, and the obtained results confirm that some coupling is a good indicator of change impact. All produced models can be integrated within a knowledge-based architecture.*

**Keywords:** *change impact, coupling, machine-learning, knowledge-based models.*

## 1. Introduction

The big issue in software engineering is to produce high quality software in time and in budget. To pass through this issue, measurement based on metrics should be done at different stages of the software development life cycle, with objectives as, software quality prediction by learning from past experiences. In fact, software metrics provide a quantitative approach to control the software development and then the software products quality. They are also a crucial source of information for decision-making. A large number of object-oriented (OO) measures have been

proposed in the literature. A particular emphasis has been given to the measurement of design artefacts, in order to help assess quality early on during the development process. Such measures are aimed at providing ways of assessing the quality of software. Such an assessment of design quality is objective, and the measurement can be automated. But how do we know what measures actually capture important quality aspects? As it is stated by the ISO/IEC international standard (14598), internal metrics are especially helpful when they are related to external quality attributes, e.g., maintainability, reusability, usability, etc.

Through various studies, maintainability has been expressed as the difficulty to correct faults or faulty components, fault-proneness, modification impact, etc. This last expression of maintainability is that which interests us in the present study. Systems modification is a difficult task that has an impact on systems becoming [3]. Change effects must be considered; a small change can have considerable and unexpected effects on the system. When modularity is adequately used, it limits the effects relating to changes. Nevertheless, change impacts are subtle and difficult to discover; designers and maintainers need mechanisms to analyze changes and to know how they are propagated in the whole system.

The ultimate motivation of our work is to improve the maintenance of OO systems, and to intervene more specifically on change impact analysis. By identifying the potential impact of a modification, one reduces the risk to deal with expensive and unpredictable changes. Consequently, we try to give more explanations on real and responsible factors for change impact and its evolution. To reach this objective, we explore several predicting or assessment models proposed by Artificial Intelligence (AI).

In this paper, section 2 presents various works related to different expressions of maintainability, and specifically to

change impact. The hypothesis concerning change impact is then stated in section 3. Section 4 concerns the building and learning of predictive models, following various Machine-Learning (ML) approaches. Moreover, we illustrate how to use these models to predict change impact for OO applications. Finally, the results are discussed, and our work perspectives are presented in the conclusion.

## 2. Related work

Several studies were conducted to validate metrics and to relate them to some properties like maintainability. Li and Henry [1] took five metrics of Chidamber and Kemerer [2], added three of their own, to show that there is a strong relationship between these metrics and maintenance effort, expressed in number of changed lines code. In [4], the authors showed that the choice of architectures, in early stages of software systems design, has an important impact on a number of quality factors, for instance, maintainability, efficiency, and reusability. In [5], Lounis & al. proposed a succession of 24 code metrics to generate predictive models related to fault-proneness, an other expression of maintainability. Moreover, in [6], the authors also studied relationships between design coupling, i.e, most of the coupling metrics, cohesion, and inheritance ones in one side, and classes' fault-proneness in the other side.

Concerning maintainability expressed as change impact. Han [7] developed an approach for computing change impact on design and implementation documents. In [8], Antoniol & al. predicted evolving object-oriented systems size starting from the analysis of the classes impacted by a change request. They predicted changes size in terms of added/modified lines of code. In an other work, Kung and al [9], interested by regression testing, developed a change impact model based on three links: inheritance, association, and aggregation. They also defined formal algorithms to calculate all the impacted classes including ripple effects. Lee and Offutt examined in [10], the effects of encapsulation, inheritance, and polymorphism on change impact; they also proposed algorithms for calculating the complete impact of changes made in a given class. On the other hand, Briand and al., in [6], tried to see if coupling measures, capturing all kinds of collaboration between classes, can help to analyze change impact. Lastly, in [11] and [12], a change impact model was defined at an abstract level, to study the changeability of object-oriented systems. The adopted approach uses characteristic properties of object-oriented systems design, measured by metrics, to predict changeability.

Many different approaches have been proposed to build such empirical predictive models; for example, they can be mathematical models (case of statistical techniques like linear and logistic regression) [1] [13], or artificial intelligence-based models (case of machine-learning techniques). In all cases, they allow affecting a value to a

quality characteristic based on the values of a set of software measures, and they allow the detection of design and implementation anomalies early in the software life cycle. They also allow organizations that purchase software to better evaluate and compare the offers they receive. As far as we know, Porter and Selby [14] have been the first to use a ML algorithm to automatically construct software quality models. They have used a classification algorithm, to identify those product measures that are the best predictors of interface errors likely to be encountered during maintenance. After Selby & Porter, many others, e.g., [15], [16], have used classification algorithms to construct software quality predictive models. More recently, [17] have investigated ML algorithms with regard to their capabilities to accurately assess the correctability of faulty software components. On the other hand, in most above-mentioned techniques, the estimation process depends on threshold values that are derived from a sample base. Thus, in a previous study, we also worked on the identification of trends instead of the determination of specific thresholds by replacing them with fuzzy thresholds [18]. In [19], [20], and [21], Fenton and Neil present the advantages of a causal-modelling approach using Bayesian networks compared to a naive regression-based approach. They also prove through case studies that Bayesian nets can provide relevant predictions, as well as incorporating the inevitable uncertainty, reliance on expert judgement, and incomplete information that are pervasive in software engineering.

In the present work, we explore how non probabilistic and probabilistic approaches could help predicting maintainability expressed as change impact. The following section presents the studied hypothesis: it concerns how coupling could influence change impact.

## 3. Change impact hypothesis proposal

For leading our work, we follow a three-step process: (i) hypothesis proposal, including metrics selection, (ii) predictive models generation, and (iii) models evaluation and use.

We are interested by the relationship between inter-classes dependencies, namely coupling, an architectural property, and change impact. Our objective is to see which types of coupling influences more change impact. In our domain literature review, we have noticed that very few works propose a more or less complete definition of change impact, i.e., model taking into account main links that one can find in an object-oriented design (namely, association, aggregation, invocation, and inheritance). In our study, we took the impact model defined in the SPOOL project [22]; we consider it as one of the most general. It allows impact calculation in a systematic way; this is an important factor concerning effort and maintenance cost reduction.

When a change is considered, it is necessary to identify system components that will be impacted; it will ensure that the system will still run correctly after change implementation. Our concern is then focused on how the system reacts to a change. It is generally accepted that a system absorbs easily a change if the number of impacted components is small. A component refers to a class, a method, or a variable. As examples of changes, one can have the deletion of a variable, the change in a method's scope from "public" to "protected" or the removal of the relationship between a class and its parent. A total of 13 changes are identified. In this context, we call change impact the set of classes that require a correction after this change. In our work, we are interested only in changes that have a syntactic impact; a given change is characterized by a code transformation somewhere in the system. If the system is successfully re-compiled, then there is no impact; otherwise, we have an impact. In [23], the authors give the final list containing a total of 52 changes, including 12 changes for variables, 25 for methods, and 15 for classes.

On the other hand, we worked with a set of metrics related to coupling. They are presented in table 1.

Metrics	Definition
RFC	Response For a Class: number of methods called upon in response to a message.
MPC	Message Passing Coupling: number of messages sent by a class in direction of the other classes of the system.
CBOU	CBO Using: refers to the classes used by the target class.
CBOIUB	CBO Is Used By: refers to the classes using the target class.
CBO	Coupling Between Object: number of classes with which a class is coupled.
CBONA	CBO No Ancestors: CBO without considering the classes ancestors.
AMMIC	Ancestors Method–Method Import Coupling: number of parents classes with which a class has an interaction of the method-method type and a coupling of the type Import Coupling (IC).
OMMIC	Others Method–Method Import Coupling: number of classes (others that super classes and subclasses) with which a class has an interaction of the method-method type and a coupling of the type IC.
DMMEC	Descendants Method–Method Export Coupling: number of subclasses with which a class has an interaction of the method-method type and a coupling of the type Export Coupling (EC).
OMMEC	Others Method–Method Export Coupling: number of classes (others that super classes and subclasses) with which a class has an interaction of the method-method type and a coupling of the type EC.

**Table 1.** The selected coupling metrics

The next section will focus on the applicability of ML algorithms for predictive models generation and use. Our objective is to produce efficient and usable models, for predicting change impact in OO applications.

#### 4. Building and using predictive models

Machine-learning is of a significant help for the building of predictive models. It is an important and prolific sub-field of AI; it proposes many approaches like induction, deduction, analogy, probabilistic, soft-computing, etc. In this study, we will see how some of them could produce efficient change impact predictive models; then, we will discuss their utility in a context where they are integrated in an automated decision-making solution.

Rules, decision trees, and Bayesian Networks (BN) are interesting representation models. Thus, rules are a natural and elegant way to model an expert knowledge; they are also the first representation choice for most of the solutions based on automated tools. Concerning decision trees, most of the algorithms producing them, are very efficient, and models induced by these algorithms are also efficient. Finally, Bayesian networks constitute a particular quantitative approach which can integrate uncertainty within reasoning, offering thus explanations that are close to reality. Moreover, with BNs, it is also possible to exploit experts' judgements to anticipate predictions. In addition, BNs have the capacity of incremental training on data; this is true as well for parameters training as for structure training, facilitating the model evolution. This characteristic will contribute to the improvement of Bayesian network structure and parameters, by the acquisition of new data.

We have selected several algorithms belonging to various ML approaches, and we have run them on software data collected from a Java medium size application. We chose a program analysis toolbox system, called BOAP [24]. It is a set of integrated software tools, which allow an expert to evaluate some software qualities, e.g., conceptual or structural weaknesses, too complex instructions, etc. We considered the BOAP system in its version 1.1.0; it is written in Java and contains 394 classes. The metrics considered in this work (see table 1) are extracted from this system.

Some of the used ML algorithms are implemented in WEKA, an open source data-mining environment [25]. We have selected them in respect to three approaches. The first one is the induction of decision trees approach; it is represented by J48, an implementation of the well-known C4.5 algorithm [26]. It is a supervised learning algorithm that induces a classification model represented by a decision tree (or equivalent rules). The second approach is rules induction; Jrip, PART, and CN2 represent this approach. The former implements a propositional rule learner. It grows rules by greedily adding conditions with

highest information gain, and after that, proceeds to incrementally prune each rule. On the other hand, PART allows the induction of rules by the iterative generation of partial decision trees; its main idea is to build a partial decision tree instead of an entirely explored one. Finally, CN2 [27] implements the covering technique, which represents classification knowledge as a disjunctive logical expression defining each class. The last approach is a hybrid one; it is illustrated by NBTre (Naïve-Bayes decision-Tree) that combines naive Bayesian classifiers and classifiers based on decision trees. It exploits a tree structure to divide the instances space into subspaces and to generate a naive Bayesian classifier for each subspace.

The computation of models accuracy is done thanks to a cross-validation procedure. It is helpful when the amount of data for training and testing is limited, which is our case; thus, every instance has been used exactly once for testing. Table 2 resumes the best obtained accuracies, for each ML algorithms category.

Change impact – Coupling measures	
Induction of decision trees: J48 or C4.5	73.85%
Induction of rules: Jrip, PART, or, CN2	68.27%
Induction of Bayesian decision trees: NBTre	66.75%

**Table 2.** Computed accuracies

In terms of accuracy, we obtain pretty interesting results. They demonstrate the link between coupling and maintainability expressed as change impact. These results can be improved with an attribute pre-selection; we have noticed that some results are better than without the attribute selection. So, it will be interesting to repeat all the experiments with a prior attribute selection.

In terms of selected internal metrics, import coupling seems to influence much more change impact than other types of coupling, since in most cases, the impact is mainly related to this type of coupling. On the other hand, it turns out that for the classes for which the number of static methods invocations, as well as the number of classes used with the target class, is large, import coupling (measured by the AMMIC metric) determines change impact. NBTre results added more details, and showed that coupling measured by CBONA and CBOU metrics also influences the impact. Of course, more experiments on more data extracted from various and representative systems are needed to confirm such conclusions.

This is an example of rules produced by such models, and thus, usable by automated decision-making systems dedicated to software quality assessment:

Rule 1: CBONA ≤ 3.5      Rule 2 : CBONA > 3.5  
           CBOU ≤ 0.5            CBOU > 36.5  
           → impact: Weak (0.46)      → impact: Strong (0.48)

**Figure 1.** Induced rules

Another popular model is Bayesian networks which integrate uncertainty within reasoning, offering a modeling that is close to reality. BNs are the result of a merging between graph theory and probability theory. BNs are based on the Bayes theorem. This theorem describes the relations which exist between simple and conditional probabilities. If A and B are two events and if we know the probability of A, of B and B knowing A, the Bayes theorem allows to determine the probability of A knowing B:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

A BN is a causal graph where:

- Nodes represent random variables. A random variable has some states, for example “Yes” and “No”, and a distribution probability for these states, where the sum of probabilities of all states must be equal to 1.
- Oriented edges define causal relations between nodes. An edge goes from a parent node towards a child node. Parent nodes which affect the same child node must be independent variables. Each node is related to a Node Probability Table (NPT), which models uncertain relation between the node and its parents. If a node has no parent, a probability table would be associated for this node. Usually, NPTs are generally created by using a mixture of empirical data with experts judgement. In this causal graph, the cause and effect relationships between the variables are not deterministic, but probabilistic. Thus, observation of a cause or several causes doesn’t involve systematically the effect or effects which depend on them, but modifies only the probability of observing them. The particular interest of BNs is that they consider both experts knowledge (in the graph or its structure) and experiments contained in data (parameters).

The main stages of our approach for considering BNs, are the following:

- 1- Network structure construction starting from practical knowledge (empirical studies)
- 2- Parameters affectation (node probability table, fuzzy logic)
- 3- Bayesian inference (algorithms, tools)

### Network structure construction

Generally, a BN construction is done in two stages: producing the suitable graph structure, then assigning probability values to network nodes [28]. The affectation of these values is done according to domain experts or starting from empirical studies.

We checked earlier in this section, the hypothesis claiming that coupling influences change impact in an object-oriented application. However, if we consider at the same time all metrics measuring the various facets of coupling between classes, the BN construction is likely to

be hard and its structure complex. In addition, the results obtained by previous ML algorithms, affirm that among the ten selected metrics (see table 1), measuring this architectural property, five metrics are effectively relevant to change impact. Rules in figure 1 are an illustration of these relevant metrics. Some of these metrics are regarded as design metrics (*AMMIC* and *OMMIC*), others are considered as implementation metrics (*MPC*, *CBOU*, and *CBONA*). The figure 2 above presents the graph expressing this knowledge in the form of a BN. Let us note that in such a BN, the relation between parents and child nodes are causal (case of *Impact* node) or definitional (case of *DesignMetrics* and *ImplementationMetrics* nodes).

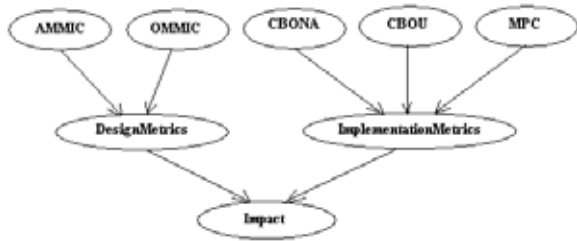


Figure 2. Change impact network

### Parameters affectation

To affect probabilities to the nodes, it is necessary to distinguish two types of variable in BN: entry variables and intermediate variables. The entry nodes probabilities are directly deduced from measurements of these variables starting from our target application BOAP.

**Entry nodes.** In our network (figure 1), the entry nodes represent the different metrics. All these entry variables are quantitative variables which have measurable numerical values. The number of possible values for these variables can be infinite; it depends of course on the considered target application. In order to facilitate the probabilities definition, these variables are initially transformed into discrete variables having a limited number of values. This transformation can be accomplished by application of fuzzy logic. Indeed, the fuzzy partitioning process replaces the various values of a metric by a set of functions which represent the membership degree (or adhesion) of each value to the various fuzzy labels (often “small”, “average”, and “large”). The fuzzy partitioning generalizes the regrouping methods by groups allowing a value to be partially classified in one or more groups at the same time. The adhesion or the value membership is distributed in all groups. However, empirically, we can determine the optimal number of groups with statistics known under the name of Dunn partition coefficient  $F_k$ . This coefficient indicates us how to gather with a better way a data set in various groups [29]. We used for that the statistics software S-plus (version 8.0) [30].

Table 3 gives an example of NPT for *AMMIC* node. It is about an example of value measured (equal to 25) for the *AMMIC* metric. To this value correspond two membership degrees (0.4349 and 0.5650) in the two fuzzy subsets. These membership degrees constitute the probabilities which are used to define the NPT of *AMMIC* node.

Small	0.43
Large	0.57

Table 3. The NPT of *AMMIC* entry node

**Intermediate nodes.** Intermediate nodes are not directly measurable. They are defined or influenced by their parent nodes. Intermediate nodes have an associated probability table. These probability values can be adjusted by using machine learning starting from the sample data or the treated cases. A parent can influence positively or negatively his child nodes. The probability distributions are affected according to the importance or the weight of each parent for the child node. At the beginning, to derive NPT, it is necessary to consider the weight of each parent node in definition or influence of its child node. For that, NPTs are initially given starting from studies in the field and from experts opinions. For instance, the *DesignMetrics* variable is defined by its two parents *AMMIC* and *OMMIC*. It is a question of finding the conditional probability of *DesignMetrics* node:  $p(\text{DesignMetrics} | \text{AMMIC}, \text{OMMIC})$ . However, like the relation between the parent nodes *AMMIC* and *OMMIC* and their child node *DesignMetrics* is definitional, the strong presence of these metrics also defines the strong presence of *DesignMetrics*. A possible scenario for the *DesignMetrics* node NPT is presented in table 4:

AMMIC	Small		Large	
	Small	Large	Small	Large
Yes	0.2	0.4	0.4	0.8
No	0.8	0.6	0.6	0.2

Table 4. The *DesignMetrics* intermediate node NPT

A reasoning which can be applied is the following: if the number of classes (others that super-classes and subclasses) with which this class has an importation interaction of the method-method type is small (*OMMIC* small), and the number of parents classes with which this class has an importation interaction of the method-method type is small also (*AMMIC* small), the design metrics presence probability in such a system is weak or small. Therefore, the probability of the state “Yes” in the probability table of *DesignMetrics* node can be 20%. It is important to recall here that there are obviously other metrics (other than those considered in this study) and which are defined like design metrics or implementation metrics, and consequently, can positively or negatively influence change impact.

**Impact node.** It corresponds to a discrete variable having three values. To define the *Impact* node NPT, we used a cross-validation process on a decision trees induction algorithm. It is a technique where the data set is divided into N blocks. A model is learned on N-1 blocks, then tested on the remaining block. The algorithm repeats the same thing for each one of N data blocks; thus the whole operation is made N times. In our case, we divided the data set (BOAP system) into 10 blocks. Table 5 gives the obtained probabilities. These probabilities are computed by considering the quantitative information associated to the leaves of the induced decisions trees.

ImplementationMetrics	Yes		No	
DesignMetrics	Yes	No	Yes	No
Weak	0.19	0.47	0.46	0.93
Average	0.09	0.07	0.12	0.04
Strong	0.72	0.46	0.42	0.03

**Table 5.** The *Impact* node NPT

### Bayesian Inference

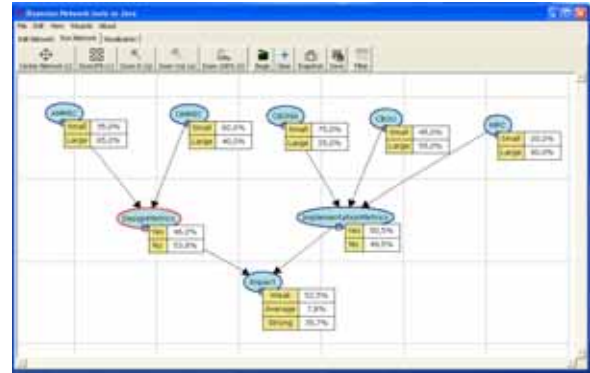
Once the graph structure and all NPTs are defined, we can proceed with the Bayesian inference. It results an update of conditional probabilities of all nodes. We have used the BNJ (Bayesian Network tools in Java) environment to achieve this goal. BNJ is a set of open source software tools intended for research and development by using graphic probabilities models. It is written in Java and is available on the web<sup>1</sup>.

Let us recall that our experimentation was made on the BOAP system (version 1.1.0) which contains 394 classes (instances). For the network execution, we randomly choose an instance from which we take the metric values corresponding to entry nodes. As soon as the probabilities distributions are updated for introduced values, we have an estimate in the form of probability for the various states assigned to the *Impact* node (see figure 3).

Having affected three states «Weak», «Average», and «Strong» to the *Impact* node, and with the used input data (see figure above), we can conclude that the change impact has a probability of 39.7% to be «Strong».

Moreover, Bayesian networks offer the possibility of processing scenarios of the form «What will occur if ...?», allowing to identify potential problems and actions to be undertaken for improvement.

A second scenario shows that by decreasing the metrics values *CBONA* and *CBOU*, change impact weakens more (its probability of being «Weak» grows from 52.5% to 60.8%). Conversely, the scenario 3 execution shows that by increasing the *CBONA* and *CBOU* metrics values; the change impact becomes increasingly strong. The probability of the «Strong» state moves from 31.5% to 46.6%.



**Figure 3.** Change impact network after scenario 1

Results obtained in the second and third scenarios confirm those already found by using a non probabilistic approach (see respectively rule 1 and rule 2 of figure 1). For example, the scenario 3 result expressing that *CBONA* and *CBOU* metrics influence positively change impact, corresponds to the result illustrated by the rule 2 presented earlier in this section.

### 5. Conclusion

The main objective of this work is to improve the quality of software products, by learning from past projects, and by capturing knowledge that will guide future developments. In this work, we have tried to show that machine learning is a feasible approach for change impact prediction; we proposed various ML approaches to analyze and predict change impact in object-oriented systems. A thorough study and a general synthesis of various former works dealing with this subject were initially essential. To verify our approach, we took a correlation hypothesis between coupling and change impact. The experimentation was made on an application containing 394 classes. The first important conclusion is relative to the accuracies obtained by the machine-learned predictive models: they are comparable for those obtained with other techniques. The results, in terms of induced models (metrics selected by decision trees and rules), were useful for the Bayesian network structure construction. Thereafter, we defined the NPTs for entry nodes, intermediate nodes, and the impact node. We used fuzzy logic to derive probabilities values starting from a set of measures (variables values or entry nodes). The network execution and the creation of several scenarios enabled us to make predictions on change impact. The results of such scenarios confirmed those already found with other non probabilistic approaches.

The main strength of such ML produced models is that we can incorporate them in a decision-making process, where, knowledge-based system architecture keeps a good separation between what we consider as an expert knowledge (the produced models, e.g., rules, trees, BNs, ...) and the procedures that exploit this knowledge.

<sup>1</sup>. <http://bnj.sourceforge.net/>



However, they have to be confirmed and generalized by more experiments on more software data, extracted from various and representative applications. This is the challenge to produce relevant and reusable models.

Finally, we are in the process of considering further experiments on other systems by including other coupling measurements, other architectural properties, and other factors which could supplement or better explain this causality relation.

## References

- [1] W. Li, S. Henry: Object-Oriented Metrics that Predict Maintainability. *J. Systems and Software*, 23 (2), 1993, 111-122.
- [2] S.R. Chidamber, C.F. Kemerer: A Metrics Suite for Object-Oriented Design. *IEEE Transactions on Software Engineering*, 20 (6), 1994, 476-493.
- [3] N. Wilde, R. Huit, "Maintenance support for object-oriented programs" in *IEEE Transactions on Software Engineering*, Vol. 18, Issue 12, Pages 1038-1044, Dec 1992..
- [4] L. C. Briand, S. J. Carrière, R. Kazman, J. Wüst: A Comprehensive Framework for Architecture Evaluation. *International Software Engineering Research Network Report ISERN-98-28*.
- [5] H. Lounis, H.A. Sahraoui, W.L. Melo: Defining, Measuring and Using Coupling metrics in Object-Oriented Environment. In *SIGPLAN OOPSLA'97 Workshop on Object-Oriented Product Metrics*, Atlanta, USA, 1997.
- [6] L.C. Briand, J. Wust, H. Lounis: Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs. In *Empirical Software Engineering, an International Journal*, March 2001, Kluwer Academic Publishers, 6(1):11-58.
- [7] J. Han, "Supporting Impact Analysis and Change Propagation in Software Engineering Environments" in *Proceedings of the STEP'97*, London, England, pages 172-182, July 1997.
- [8] G. Antonioli, G.Canfora, A. D. Lucia, "Estimating the size of changes for evolving Object-Oriented Systems : a Case Study" in *Proceedings of the 6th International Software Metrics Symposium*, pages 250-258, Boca Raton, Florida, Nov 1999.
- [9] D.C. Kung, J. Gao, P. Hsia, J. Lin, Y. Toyoshima, "Class firewall, test order, and regression testing of object-oriented programs" in *Journal of Object-Oriented Programming*, Vol. 8, No. 2, pages 51-65, May 1995.
- [10] L. Li, A. J. Offutt: Algorithmic Analysis of the Impact of Changes to Object-Oriented Software. In *proceedings of ICSM'96*, 1996, 171-184.
- [11] M.A. Chaumon, H. Kabaili, R.K. Keller and F. Lustman. "A Change Impact Model for Changeability Assessment in Object-Oriented Software Systems". In *Proceedings of the Third Euromicro Working Conference on Software Maintenance and Reengineering CSMR'99*, pages 130-138, Amsterdam, The Netherlands, March 1999.
- [12] H. Kabaili, R.K. Keller, F. Lustman, and G. Saint-Denis. *Class Cohesion Revisited: An Empirical Study on Industrial Systems*. In *Proceedings of the Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, pages 29-38, Cannes, France, June 2000.
- [13] T.M. Khoshgoftaar, J.C. Munson: Predicting Software Development Errors Using Software Complexity Metrics. In *IEEE journal on selected Areas in Communications*, vol.8, n.2, February 1990.
- [14] A. Porter, R. Selby: Empirically guided software development using metric-based classification trees. In *IEEE Software*, March 1990, 7(2):46-54.
- [15] V. Basili, Condon, K. El Emam, R. B. Hendrick, W. L. Melo: Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components. In *Proc. of the IEEE 19<sup>th</sup> Int'l. Conf. on S/W Eng.*, Boston, 1997.
- [16] M. Jorgensen: Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. In *IEEE TSE*, August 1995, 21(8):674-681.
- [17] M.A. De Almeida, H. Lounis, W. Melo: An Investigation on the Use of ML Models for Estimating Software Correctability. In *the Int. Journal of Software Engineering and Knowledge Engineering*, October 1999.
- [18] H.A. Sahraoui, M. Boukadoum, H. Lounis: Building Quality Estimation models with Fuzzy Threshold Values. In "L'objet", volume 7, number 4, 2001.
- [19] N.E. Fenton and M. Neil, "The Jury Observation Fallacy and the use of Bayesian Networks to present Probabilistic Legal Arguments", *Mathematics Today (Bulletin of the IMA)*, 36(6), 180-187, 2000.
- [20] N.E. Fenton and M. Neil, "Making Decisions: Using Bayesian Nets and MCDA", *Knowledge-Based Systems* 14, 307-325, 2001.
- [21] Neil M, Fenton NE, Nielsen L, "Building large-scale Bayesian Networks", *The Knowledge Engineering Review*, 15(3), 257-284, 2000.
- [22] R. Schauer, R. K. Keller, B. Laguë, G. Knapen, S. Robitaille, G. Saint-Denis: *The SPOOL Design Repository: Architecture, Schema, and Mechanisms*. In Hakan Erdogmus and Oryal Tanir editors, *Advances in Software Engineering. Topics in Evolution, Comprehension, and Evaluation*. Springer-Verlag, 2001.
- [23] M.K. Abdi, H. Lounis, H. Sahraoui: Analyzing Change Impact in Object-Oriented Systems. In *proceedings of the 32nd EUROMICRO Software Engineering and Advanced Applications Conference*, Cavtat/Dubrovnik (Croatia), August 29-September 1, 2006.
- [24] E. Alikacem, H. Snoussi, "BOAP 1.1.0 : Manuel d'utilisation", CRIM, January 2002.
- [25] I.H. Witten, E. Frank: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann Publishers, San Francisco, California, 2000.
- [26] J.R. Quinlan: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
- [27] P. Clark, T. Niblett: The CN2 induction algorithm. In *ML Journal*, 1989, 3(4):261-283.
- [28] P. Naïm, P. Wuillemin, P. Leray, O. Pourret, A. Becker, "Réseaux bayésiens", Edition Eyrolles, 2004.
- [29] E. Trauwaert, On the meaning of Dunn's partition coefficient for fuzzy clusters, *Fuzzy Sets and Systems*, Vol.25, No 2, pp 217-242, 1988.
- [30] *Insightful Corporation*, Seattle, WA, S-PLUS® 8 for Windows® User's Guide, Copyright © 1987-2007.

# Program File Bug Fix Effort Estimation Using Machine Learning Methods for OSS

Syed Nadeem Ahsan, Javed Ferzund and Franz Wotawa  
Institute for Software Technology  
Technische Universität Graz  
8010 Graz, Inffeldgasse 16b/2, Austria  
{sahsan, jferzund, wotawa}@ist.tugraz.at

**Abstract**-Accurate effort estimation model plays an important role in software maintenance and software project management. Most of the effort estimation models are related to commercial or closed software systems, whereas it is difficult to develop an effort estimation model for open source software system (OSS). Reasons may be the inherent complexity of OSS, the large number of software developers or contributors and the absence of effort data. Most of the OSS systems do not maintain effort data, while this data is required to develop and validate the effort estimation model. In this paper we present the bug fix effort estimation model for open source software system. This paper is divided into two parts; in the first part we present a heuristic approach to mine the effort data from the developers or contributors activity logs. In the second part, we present six different effort estimation models, which are based on statistical regression and machine learning (ML) methods. To develop the effort estimation model, we used a set of metrics as estimators along with the bug fix effort data, which is obtained in the first part. The set of metrics are obtained from program files, source code changes and CVS log. To perform experiments we selected the Mozilla open source project and downloaded bug fix reports along with bug fix activity data from the corresponding bugzilla server. We also downloaded source files revisions and CVS log data from CVS repository. Furthermore, we compared the outcome of the different effort models using several evaluation criteria. The results show that the machine learning models are better compared to the statistical regression model. While in case of machine learning based model, the support vector machine has the lowest relative absolute error.

## I. INTRODUCTION

The main function of the software maintenance is to keep the software alive by performing three major tasks i.e. customer support, update documents and perform changes in the source code. The changes in the source code are required to remove faults, to enhance existing features or to add new features. The timely completed maintenance task makes it possible to deliver the product in time, which is the key requirement of the today's software industry and has to be based on accurate estimation model.

There are several advantages of having an effort estimator. First, it provides initial knowledge about the complexity of the product. Second, it may be used to obtain the cost of the product. Moreover, an effort estimator allows for task assignment and resource management. Most of the research work on effort estimation model is related to closed software system [2, 3, 5], while very few attempts have been made to develop an effort estimation model for OSS [12, 8, 16]. One

reason may be the absence of the effort data, because most of the OSS systems do not maintain the maintenance effort data. While effort data are required to build and validate the effort estimation models [8].

In this paper our focus is to establish an accurate effort estimation model for corrective maintenance task of OSS systems. To perform this experiment, we obtained data from the Mozilla project repository, and precisely show how the relevant data can be extracted and under which assumptions. There are many possibilities for the model, including statistical regression and machine learning methods. In order to answer the question which method would give back the best, i.e., most accurate estimator, we tested six different methods using the same data set. The models are constructed using a set of metrics as model estimator. These set of metrics are related to developer expertise, source code changes and program files. The main contributions of our work are:

- i) Developed a method to mine the bug fix effort data from the developers or contributor's log of bug fix activity.
- ii) Developed regression and ML based effort estimation models using the obtained effort data and a set of metrics.

The paper is organized as follows: In Section 2 we discuss related work. In Section 3 we describe how we obtained the data from the software repositories. Furthermore, we explain the used program file change metrics. In Section 4 we discuss the estimation models and analyze the obtained results. Finally, we conclude the paper and discuss future work.

## II. RELATED WORK

Different approaches are used to establish estimation models, like algorithmic, analogy, hybrid and machine learning. Initially algorithmic estimation methods were used for software estimation, like Boehm's constructive cost model COCOMO [4] and COCOMO II [3], Albrecht's function point method [2] and Putnam's software life cycle management (SLIM) [15]. These are based on historical data of effort. These approaches involve the construction of mathematical models from empirical data.

Eick et al. [14] worked on the software evolution data of fifteen years. They showed that the code decays, means if the code life is large then it needs more effort to add new changes. They extracted a large number of features from the evolution data and constructed multiple models. Their regression based

effort estimation model shows that more effort is required to make changes in the older source code. Their model also shows that the number of added or deleted lines has less impact on effort. De Lucia et al. [9] obtained a data set from five different projects. They used multiple linear regression to build the estimation models. They found that the performance of their models was enhanced if they included the different types of maintenance task into their models. They used cross validation to assess their models.

Magne Jorgensen [10] developed eleven different effort prediction models based on regression, neural network and pattern recognition and reported that the model based on regression and pattern recognition are best compared to other models. Song et al. [13] used the NASA's SEL defect data set and applied association rule mining on the data set, to classify the effort using intervals. They found that association rule mining technique is better to predict the effort compared with other machine learning techniques like PART, C4.5 and Naïve Bayes. Gary D. Boetticher [5] used neural network to develop an effort estimation model. He used different combination of product metrics to train the model. His result shows that neural network can be used for an effort estimation model.

Stefan Koch [12] discussed the issue of programmer participation and effort modeling for OSS. He worked on the GNOME project data and estimated the effort on the basis of programmer participation and the product metrics. He showed that the impact of programmer participation on effort estimation is less compared to the product metrics. Cathrin Weiss et al. [16] used the available effort data of the JBoss project, which is maintained by JIRA bug reporting system. They used the text similarity and the nearest neighbor approach, and obtained the average effort data of all the resolved bugs whose summary and title are similar to the new bug report summary and title. They used the obtained average effort value as the predicted effort value of the new bug report.

Liguo Yu [8] analysed an evolution data set of 121 revisions of Linux to develop an effort estimation model for OSS. Since Linux does not maintain effort data, therefore he first performed an experiment on NASA SEL database, which is a closed software system and maintained actual effort data. From this experiment he identified those measures which can be used indirectly to represent maintenance effort. In the next step he used those indirect measures as effort and developed two regression based estimation model for Linux project. Our work is similar to his work but we used different approach to obtain the effort data. Also we used both multiple linear regression and machine learning methods to develop effort estimation model.

### III. OBTAINING DATA FROM REPOSITORIES

In order to develop an effort estimation model we have to obtain the relevant data from the available repositories. These include the metrics data and an estimate of the effort needed to fix a certain bug. In particular we rely on the Mozilla CVS repository and Mozilla bug database. In case of another

software project where a CVS repository or a bug database like bugzilla is available, all the information extraction described in this section can be directly applied. In other cases the described metrics and effort data may be extracted in a different way. However, the underlying concepts including which metrics to use and how to obtain the effort estimates can be re-used. The overall data extraction process for effort estimation is shown in Figure 1.

#### A. Metrics Data

For obtaining the metrics we downloaded the selected revisions of the C++ program files and bug reports from the Mozilla CVS and bugzilla repositories and stored them on a local disk. In the following paragraph we describe the process which we used to extract the metrics data. The complete list of metrics with description is shown in Table 1.

First we identify those revisions of program files where bugs were fixed. To accomplish this task we used an approach [6] and parsed the CVS log comments of each source file revisions. If the comment contains a word like *Bug*, *Fix* or *Fixed* followed by some integer value, which is similar to any of the existing bug report id. Then it means that revision is a bug fixed revision. After identifying all the bug fixed revisions, we extracted all those lines of code from bug fixed revisions, which have been changed to fix the bug. To perform this task we take the differences between two consecutive revisions, i.e., the revision where the bug has been fixed and its immediate predecessor. These program files difference data are further processed to obtain a set of metrics related to program file changes i.e.  $T_{CLOC}$ ,  $T_{DELTA}$  and  $T_{COPE}$ . We also processed all those program file revisions which are immediate predecessor of the bug fix revisions and extracted the set metrics related to whole program file i.e.  $T_{LOC}$ ,  $T_{FUNC}$ ,  $T_{ELINE}$ ,  $T_{FINC}$ ,  $T_{CYCLO}$ ,  $T_{PCOUNT}$ , and  $T_{RPOINT}$ . We also obtained

TABLE 1  
LIST OF METRICS

<i>Metrics</i>	<i>Description</i>
$S_{FCOUNT}$	Number of source files which are changed to fix the bug.
$D_{CCOUNT}$	Number of developers who are involved in fixing a bug.
$D_{EXP}$	Developer's Expertise
$T_{PREV}$	Total Source File Age: Adding all the previous revisions of the source files which are involved in fixing the bug.
$T_{PFREV}$	Total previous fix revisions of source files.
$T_{LOC}$	Total line of source code.
$T_{CLOC}$	Total changed line of code.
$T_{DELTA}$	Total number of change location in source files. Delta is change hunk pair, we have at least one delta whenever a file is changed.
$T_{FINC}$	Total number of included source files/packages.
$T_{FUNC}$	Total number of function.
$T_{ELINE}$	Total number of executable lines.
$T_{CYCLO}$	Total cyclomatic complexity metrics.
$T_{PCOUNT}$	Total number of parameter count.
$T_{RPOINT}$	Total number of return points.
$T_{COPE}$	Total number of changed operators.

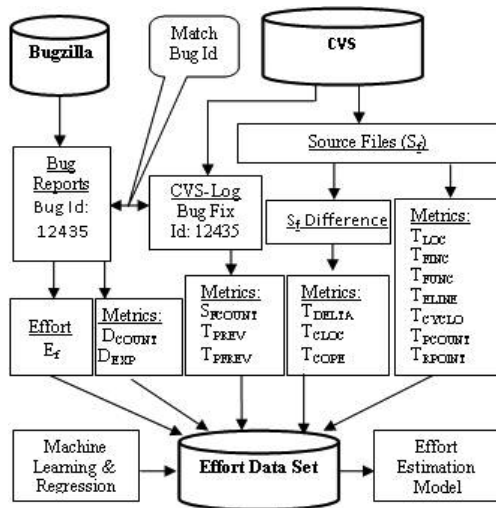


Fig. 1. Data extraction process

the total number of previous revisions  $T_{PREV}$ , the total number of program files which are changed to fix a bug i.e.  $S_{FCOUNT}$  and the total number of previous bug fix revisions  $T_{PFREV}$  of each bug fixed revision of source file.

Finally we processed bug reports and developer’s activity data which are related to the bug fixing activities, and obtained the two measures i.e. developer counts ( $D_{COUNT}$ ) and developer’s expertise ( $D_{EXP}$ ). The  $D_{COUNT}$  is the count of developers or contributors who were involved in fixing a bug. The  $D_{EXP}$  is obtained by adding the rank values of all those developers who were involved in fixing a bug. We rank the developers according to their number of bugs fix count. To perform this task we downloaded 93,607 bug reports together with the bug fix history from the bugzilla repository. For detailed description of the above mentioned process, we refer the reader to our technical report [1].

### B. Bug Fix Effort

The conventional maintenance effort estimation process involved three steps. In the first step it is required to extract maintenance effort data along with other related measures from the previous maintenance records. Then in the second step, it is required to build the model using the data obtained in the first step. While in the third and final step the model may be used to predict the future maintenance effort [8]. Unfortunately most of the OSS system does not maintain actual effort data and consequently it becomes more difficult to develop an accurate effort estimation model. However software repositories contain a lot of maintenance related data. In the previous section we have described in detail how we have extracted all the maintenance related measures from these repositories. Now in this section we describe our approach and method which we have used to extract the actual bug fix effort data from bug repository.

The most frequently used maintenance effort measure is the total number of man-hours required to accomplish the maintenance task. Therefore we focus to extract the actual time spent by developer to fix the bugs and we considered that time as an estimated actual bug fix effort.

Bug reporting systems are used in open source software to store the software maintenance records. Mozilla (<http://www.mozilla.org>) and lots of others OSS use Bugzilla (<http://bugzilla.mozilla.org>), as a bug reporting system. Bugzilla maintain the complete history of bug life cycle of all the reported bugs. Each reported bug in bugzilla has a complete life cycle. Figure 2 shows the bug life cycle. According to this life cycle, a bug starts as UNCONFIRMED. It immediately moves to the status NEW, after that the bug is validated by the quality assurance (QA) person. Then it is moved to ASSIGNED status, which is then followed by the RESOLVED status. Finally a bug may reach a status of VERIFIED, which is then followed by CLOSED. In some cases after the VERIFIED status a bug may go to REOPEN status, and the cycle is repeated.

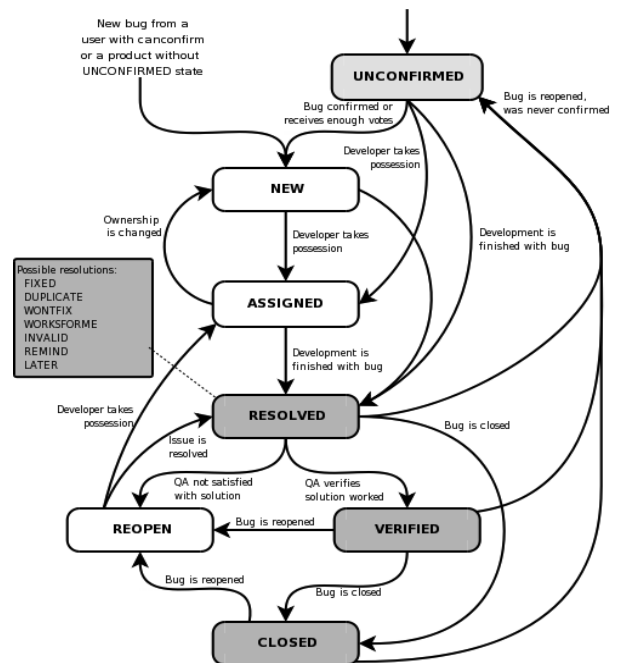


Fig. 2 The Bugzilla bug life cycle

Let us now go into more detail of the bug life cycle. If a bug is said to be NEW, the QA person assigns the bug to any relevant developer/contributor in order to find a solution. Hence, the real work for fixing a bug starts when a bug is moved to the ASSIGNED status. The bug is solved when the developer or contributor provides a solution and moves the status to RESOLVED. The duration between ASSIGNED and RESOLVED is the actual period where effort is spend on source code changes to fix the bug. Hence, we assume that this time period is the actual bug fix time and thus the only time to be considered as effort. Note that in some cases a QA person reassigns the same bug to another developer or in some cases the previously assigned developer assigns the bug to some other developer, which makes the computation of the overall effort even more difficult. Since no information regarding the distribution of effort among the different developers is available, we assume that all developers contribute. Thus we calculate the sum of all time periods for each developer or contributor to come up with a single total bug fix effort.

Developer Name: X      Month: February																																																		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28																							
										Bug Id= 1, Bug fix duration 06 days																																								
		Bug Id= 2, Bug fix duration 10 days																																																
		Bug Id= 3, Bug fix duration 21 days																																																
												Bug Id= 4, Bug fix duration 10 days																																						
		Total working days $T_w = 24$ days																																																

Fig. 3 Developer monthly bug fix activity

For a single developer we compute the effort necessary to provide a solution to a bug report as follows: We start with the bug reports assigned to the developer. We compute the effort assigned to a bug report for each month of the year using the given dates for ASSIGNED and RESOLVED. The time span between ASSIGNED and RESOLVED cannot be used directly to compute the effort. The reason is that a developer works on several bug reports in parallel but it is impossible to spend more than all days of a month in working. Hence, the time spent has to be multiplied by a factor. This factor takes into account the limited number of days available for working within a particular month. To understand how we have estimated bug fix effort from bug reports, consider an example in which a developer has fixed four bugs in the month of February. The example data is shown in Fig. 3. The gray bar represents the durations in which the developer fixed those bugs. We obtained these durations from bug reports by subtracting the bug assigned date from the bug resolved date. In this example we assume that during some days of the month the developer worked in parallel on multiple bugs. Therefore we have to multiply the duration of bug fix with a common multiplication factor ( $M_k$ ). We obtain the multiplication factor by dividing the total actual working days of a month ( $T_w$ ) with the sum of all the assigned working days for all the assigned bugs,  $M_k = T_w / \sum Days = 24/47 = 0.51$  and  $E_i = \sum (Bug\ fix\ duration\ for\ bug_i) \times M_k$ . Where  $E_i$  is the bug fix effort for bug  $i$ , and  $k$  is the number of months spent to fix a bug. In this example  $k=1$ . Therefore,  $E_1 = 6 \times 0.52 = 3.12$ . Similarly we can estimate effort for other bugs,  $E_2=5.1$ ,  $E_3=10.7$ , &  $E_4=5.1$ .

If a bug is fixed in more than one month, then the bug fix duration for the first month is obtained by subtracting the bug assigned day from the last day of the month, and for the last month, the bug fix duration is obtained by subtracting the first day of the month from the bug resolved day. We consider the whole days of a month as bug fix duration for all the intermediate months. We multiplied each month's bug fix duration with the respective multiplication factor. Finally, we add all the obtained values to get the estimated effort value.

We have applied the above mentioned method on each downloaded bug report and developer's activity data, and obtained the complete history of the previous bug fix efforts. We used this valuable data and created a log book that contains the bug fix effort history of all those developers or contributors who ever fixed at least one bug. An example of the automatically generated log book is shown in Fig. 4. Besides its main advantage of providing the effort data, there may be several other advantages, like it may be used for the

analysis of developer's activity patterns. It is shown in Fig 4 that the developer *Neil* is more active during the last months of the year 2001 as compare to the initial months of the same year, similarly we can use this log book data to analyze the month wise or year wise effort distribution.

#### IV. EFFORT ESTIMATION MODEL

The second and most important contribution of this paper is the development and comparison of different models for extracting an effort estimator from the obtained metrics and effort data. Names of all these methods are shown in Table 3. For the evaluation of models we have used the mean absolute error (MAE), the root mean square error (RMSE), the mean relative absolute error (MRE), the mean magnitude of relative error (MMRE), the root relative square error (RRSE), the correlation coefficient R, and the percentage of prediction PRED(x). Following are the formal definitions of these evaluation measures. Where  $E_{ACT}$  stands for the actual effort,  $E_{PRED}$  for the predicted effort, and  $n$  for the total number of observed values.

$$MAE = \frac{\sum |E_{ACT} - E_{PRED}|}{n}, \quad MMRE = \frac{\sum_{j=1}^n MRE_j}{n},$$

$$RMSE = \sqrt{\frac{\sum (E_{ACT} - E_{PRED})^2}{n}}, \quad RRSE = \sqrt{\frac{\sum (E_{ACT} - E_{PRED})^2}{E_{ACT}}},$$

$$MRE = \frac{|E_{ACT} - E_{PRED}|}{E_{ACT}}, \quad PRED(x) = \frac{1}{n} \sum_{j=1}^n \begin{cases} 1 & \text{if } MRE_j \leq x \\ 0 & \text{otherwise} \end{cases}$$

According to Conte et al [11], the MMRE value for the best effort prediction model should be  $\leq 25\%$ , and PRED(25)  $\geq 70\%$ . To develop estimation models, we used freely available ML tool WEKA (<http://www.cs.waikato.ac.nz>). For evaluation purposes we used different models on a dataset, which comprises 7027 number of data instances.

##### A. Multiple Linear Regression Model

To develop an effort estimation model using statistical multiple linear regression (MLR), we used a set of metrics as set of estimators for the model, and effort as a dependent variable. The set of metrics is shown in Table 2. Beside the used metrics, Table 2 also comprises the Pearson and Spearman correlation of the metrics with the effort. The obtained MLR model is given below,

$$EFFORT = -13.8 \times D_{COUNT} - 1.09 \times D_{EXP} + 5.8 \times S_{FCOUNT} - 0.01 \times T_{PREV} + 0.02 \times T_{PFREV} + 0.001 \times T_{LOC} + 0.02 \times T_{CLOC} - 0.035 \times T_{DELTA} - 0.03 \times T_{FUNC} + 0.004 \times T_{ELINE} - 0.025 \times T_{CYCLO} + 0.02 \times T_{PCOUNT} - 12.9$$

Developer Name: neil@httl.net													Log Year: 2001		
Bug Id	Date Assigned	Date Resolved	Jan	Feb	Mar	Apr	May	Jun	July	Aug	Sep	Oct	Nov	Dec	Total Days
1697	2001-04-25	2003-07-31				02.31	08.74	06.47	04.43	04.11	03.75	03.31	02.86	02.28	038.26
38367	2000-05-16	2003-04-07	15.50	21.19	27.46	13.85	08.74	06.47	04.43	04.11	03.75	03.31	02.86	02.28	113.95
54175	2000-12-28	2001-02-09	15.50	06.81	00.00										022.31
58523	2001-12-11	2005-03-02													001.47
66475	2001-10-12	2004-11-28											02.03	02.86	007.17
72481	2001-03-27	2001-08-15			03.54	13.85	08.74	06.47	04.43	01.99					039.02
75686	2001-07-27	2003-02-28							00.57	04.11	03.75	03.31	02.86	02.28	016.88
80837	2001-05-14	2003-03-20				04.79	06.47	04.43	04.11	03.75	03.31	02.86	02.28	02.28	032.00
85908	2001-06-15	2003-09-11					03.24	04.43	04.11	03.75	03.31	02.86	02.28	02.28	023.98
87924	2001-06-26	2003-09-11					00.86	04.43	04.11	03.75	03.31	02.86	02.28	02.28	021.60
89212	2001-07-04	2003-10-17						03.86	04.11	03.75	03.31	02.86	02.28	02.28	020.17
97532	2001-08-29	2005-09-30							02.60	03.75	03.31	02.86	02.28	02.28	012.46
99328	2001-10-08	2002-12-12										02.46	02.86	02.28	007.60
107418	2001-11-16	2003-01-17											01.33	02.28	003.61
110254	2001-12-02	2002-01-21												02.14	002.14
111606	2001-11-27	2001-11-28											00.10	00.00	000.10
114522	2001-12-11	2002-01-15												01.47	001.47
116196	2001-12-20	2002-09-13												00.81	000.81
Days per month used to fix the bug			31.00	28.00	31.0	30.01	31.01	29.98	26.58	31.00	30.00	28.94	29.20	30.08	365
Assigned bugs per month			2	2	2	3	4	6	8	9	8	11	12	14	

Fig. 4 Example of log book that contains the complete history of bug fix effort

The correlation data in Table 2 shows that  $S_{FCOUNT}$ ,  $D_{COUNT}$ , and  $D_{EXP}$  are positively correlated with the effort data. The metrics  $D_{COUNT}$ , and  $D_{EXP}$  are related to developers while  $S_{FCOUNT}$  is the total numbers of source files, which are changed to fix a bug. Whereas the metrics which are directly related to the source code like line of code  $T_{LOC}$ , cycloramic complexity  $T_{CYCLO}$  etc are positively correlated with the effort value, but their correlation with effort is not high. It shows that in case of OSS, metrics related to developers are more correlated with effort as compared to the source code metrics. The obtained Pearson's correlation coefficient value of the model is 0.53, this indicates that the predicted effort value using MLR model is highly correlated with the actual effort values. While  $R^2$  and adjusted  $R^2$  values are 0.289 and 0.291 respectively. The further results of model accuracy estimation are shown in Table 3.

### B. Machine Learning Models

In order to develop an accurate effort estimation model, we have analyzed several machine-learning (ML) algorithms using WEKA tool [7]. All the selected ML algorithms belong to the class of supervised learning methods, which are commonly used for classification and regression. To obtain a better model we used 10 fold cross-validation. It is an important technique to avoid over-fitting models on training data, as over-fitting will give low accuracy on validation. It actually divides the data set into 10 equal parts and randomly selects 9 parts for training and 1 part for testing and repeats it for 10 times [7]. In the following paragraph we discuss each model and its performance.

To obtain the support vector machine (SVM) based model, we used SVMreg algorithm, and we used the filter data type i.e. normalized training data. While to obtain a neural network based model, we used multilayer perceptron, which belongs to the feed forward class of networks. We designed multiple neural networks using 1, 2, and 3 hidden layers with 2, 4-2 and 6-4-2 perceptron per hidden layer, and set the number of learning steps between 500-1000. For M5Rules

TABLE 2.  
CORRELATION OF METRICS WITH EFFORT

Sr.	Metrics	Correlation With Effort		Mean	Standard Deviation	p-value (sig-level 0.001)
		Pearson	Spearman			
1	$S_{FCOUNT}$	0.32	0.51	1.8	1.6	0.000
2	$D_{COUNT}$	0.41	0.43	2.2	1.5	0.000
3	$D_{EXP}$	0.36	0.36	15.2	10.1	0.000
4	$T_{PREV}$	0.19	0.24	224.4	289.4	0.000
5	$T_{PREFEV}$	0.18	0.28	144.1	199.5	0.000
6	$T_{LOC}$	0.20	0.27	2472.0	2799.3	0.000
7	$T_{CLOC}$	0.18	0.30	39.4	85.5	0.000
8	$T_{DELTA}$	0.22	0.23	10.0	20.9	0.000
9	$T_{FINC}$	0.21	0.32	52.8	59.7	0.000
10	$T_{FUNC}$	0.21	0.28	103.3	115.3	0.000
11	$T_{ELINE}$	0.19	0.27	2200.8	2425.9	0.000
12	$T_{CYCLO}$	0.20	0.26	424.1	482.9	0.000
13	$T_{PCOUNT}$	0.21	0.27	179.4	200.1	0.000
14	$T_{RPOINT}$	0.18	0.24	171.3	215.4	0.000
15	$T_{COPE}$	0.13	0.24	29.3	67.4	0.000

method we used two classification rules on the basis of the  $D_{COUNT}$  metrics value. We also used the decision tree M5P and fast decision tree learner method REPTree with the pruning option. Table 3 depicts the obtained results. We see that each model has a good correlation value. The highest correlation value is for the classification rule M5Rules i.e., 0.56, while its MMRE value is 74%. In case of SVMreg the correlation value is 0.51 and the MMRE value is 63 %, which is the lowest. Therefore in our experiment the best model is the support vector machine SVMreg, although the value of MMRE is acceptable, while  $PRED(0.25) = 0.20$  and  $PRED(0.5) = 0.45$ , which is not very close to the ideal value. There are several reasons of MMRE value being so high. One may be the presence of noise or outlier in the data set. The other may be

TABLE 3  
EVALUATION RESULTS OF DIFFERENT EFFORT ESTIMATION MODELS

Sr. No	Method Name	R	MAE	RMSE	MMRE (%)	RRSE (%)	PRED(x)	
							0.25	0.50
1	Multiple Linear Regression	0.54	11.88	23.5	81.0	84.7	0.09	0.17
2	Support Vector Regression (SVMreg)	0.51	9.36	26.7	63.8	95.8	0.20	0.45
3	Neural Network (Multilayer Perceptron)	0.54	29.2	58.2	93.6	86.3	0.10	0.22
4	Classification Rule (M5Rules)	0.56	10.8	23.0	73.6	82.5	0.18	0.28
5	Decision Tree (REPtree)	0.51	10.8	23.9	74.23	86.0	0.10	0.23
6	Decision Tree (M5P)	0.55	10.6	22.9	77.7	82.5	0.13	0.26

the quality of the effort data set because we don't get it from Mozilla project rather we extract it by our own heuristic method. Another big issue with the MMRE is its value strongly influenced by a few very large MRE values [10].

#### V. THREATS TO VALIDITY

The computation of effort spent to correct a bug described in a bug report assumes that the real effort is distributed evenly. This might not be the case but since there is no other information available, it is the best we can do. This assumption as well as the others introduces some errors in the resulting data. But given the huge amount of data available in the repositories we expect that there is no error inherently built in this system of computing the effort. Hence, there might be a decrease of reliability in the data but there should always be an upper bound.

We also assumed that developers spend whole assigned period in fixing the bug, but this might not be the case, because in OSS only some experienced developers are doing as a full paid job, while most of the contributors are volunteers and they may be involved in some other jobs during bug assigned period. Also one cannot completely rule out the existence of any outliers. However we have almost removed most of them from our dataset.

#### VI. CONCLUSION AND FUTURE WORK

In this paper we presented the result obtained from different effort estimation models for OSS system. These models are based on statistical methods as well as machine-learning methods. All the models are based on the same underlying metrics and effort data and trained with the same number of instances i.e. 7027. Since most of the OSS system does not maintain the bug fix effort data, therefore we developed a method for deriving this information from bug repositories. We processed the developer's activity log data and obtained the bug fix effort values in terms of bug fix days. Table 3 shows that the machine learning and multiple linear regression based effort estimation models have correlation values between 0.51 and 0.56. Similarly the MMRE values lie between 63% and 93%. This shows that the performance of our models is satisfactory. In future we will work to identify some other metrics that have good correlation with effort data. We will continue our work on bug reports to improve the extraction of effort data. These will ultimately improve our effort estimation model.

#### REFERENCES

- [1] Syed Nadeem Ahsan, Javed Ferzund, and Franz Wotawa, "Mining Software Repositories for Software Estimation Model," Technical Report, Institute for Software Technology, TU-Graz, 2009.
- [2] Albrecht, A.J. Gaffney, J.E., Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, vol.SE-9, no.6, pp. 639-648, Nov. 1983
- [3] Boehm, B., et al., "Cost Models for Future Software Life Cycle Process: COCOMO 2," *Annals of Software Engineering*, 1995.
- [4] B. W. Boehm, "Software Engineering Economics," *Englewood Cliffs, NJ*, Prentice Hall PTR, Oct 1981.
- [5] Boetticher, G., "An Assessment of Metric Contribution in the Construction of a Neural Network-Based Effort Estimator," *Second Int. Workshop on Soft Computing Applied to Software Engineering*, 2001.
- [6] M. Fischer, M. Pinzger, and H. Gall. "Populating a release history database from version control and bug tracking systems," ICSM 2003.
- [7] Ian H. Witten, *Data Mining Practical Machine Learning Tools and Techniqu.*, Second Edition, 2005.
- [8] Yu, L. "Indirectly predicting the maintenance effort of open-source softw.: Res. Articles." *J.Softw.Maint.Evol.* 18, 5(Sep.06), 311-332, 2006.
- [9] De Lucia, A., Pompella, E., and Stefanucci, S. 2002. "Effort estimation for corrective software maintenance." In *Proceedings of the 14th international Conference on Software Engineering and Knowledge Engineering* (Ischia, Italy, July 15 - 19, 2002). SEKE '02, vol. 27.
- [10] Jorgensen, M., "Experience with the accuracy of software maintenance task effort prediction models," *IEEE Trans. Softw. Eng.* vol.21, no.8, pp.674-681, Aug 1995.
- [11] Conte, S. D., Dunsmore, H. E., and Shen, Y. E. 1986 *Software Engg. Metrics and Models*. Benjamin-Cummings Publishing Co.
- [12] Koch, S., Effort Modeling and Programmer Participation in Open Source Softw. Projects, *Inform. Economics and Policy*, 20(4): 345-355.
- [13] Qinbao Song, Martin Shepperd, Michelle Cartwright, Carolyn Mair, Software Defect Association Mining and Defect Correction Effort Prediction, *IEEE Trans. Softw. Eng.* vol. 32, no. 2, pp. 69-82, 2006.
- [14] Eick, S. G., Graves, T. L., Karr, A. F., Marron, J. S., and Mockus, A. 2001. Does Code Decay? Assessing the Evidence from Change Management Data. *IEEE Trans. Softw. Eng.* 27, 1 (Jan. 2001), 1-12.
- [15] Putnam, L.H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *Software Engineering, IEEE Transactions on*, vol.SE-4, no.4, pp. 345-361, July 1978
- [16] Weiss, C., Premraj, R., Zimmermann, T., and Zeller, A. 2007. "How Long Will It Take to Fix This Bug?." In *Proceedings of the Fourth international Workshop on MSR* (May 20 - 26, 2007). International Conference on Softw. Eng. IEEE Computer Society, Washington, DC, 1.

#### ACKNOWLEDGMENT

The research work presented in this paper is partially funded by the Higher Education Commission (HEC), Pakistan and partially conducted within the competence network Softnet Austria (www.soft-net.at) that is funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsförderungs-gesellschaft mbH. (SFG), and the city of Vienna in terms of the centre for innovation and Technology (ZIT).

# An Architecture-based Evolution Management Method for Software Product Line

Xin Peng, Liwei Shen, Wenyun Zhao  
School of Computer Science, Fudan University, Shanghai 200433, China  
{pengxin, 061021062, wyzhao}@fudan.edu.cn

## Abstract

*In software product line (SPL) development, evolutions occur in core assets and application products. How to ensure their alignment in evolution is a big challenge. Products in an SPL share a reference architecture, which centers in SPL development and evolution, so architectural evolution management is a natural and essential choice for SPL. In this paper, we propose an architecture-based evolution management method for SPL, in which both architecture and component evolutions are supported. An integrated version model for both core assets and application products is proposed. Based on the model, the method provides evolution processes for architectures and components, both supporting forward customizations and backward feedbacks by merging and synchronization. The prototype tool for the method has been developed on the open-source version control system Subversion, and preliminary application has shown that it can effectively support SPL evolutions.*

## 1. Introduction

In SPL, there are both domain-level (domain engineering) and product-level (application engineering) developments with different goals and disciplines. These two kinds of development activities are often inclined to evolve to different directions if there is no effective coordination. For example, application engineers may decide to make architectural adaptations incompatible with the reference architecture or directly modify domain components under demands of product customers. If this kind of deviations accumulates, the organization will lose the control on the product line gradually. Therefore, successful SPL engineering requires management and coordination of two kinds of development activities to meet the organization's overall business goals [1].

Software configuration management (SCM) is the discipline of managing the evolution of complex software systems [2]. The discipline enables us to keep control and track software changes, and is an integral part of any software development and maintenance activity [3]. In traditional software development, SCM is performed within each project. However, in SPL, evolution management of application-engineering projects and the domain-engineering project should be

coordinated and unified to maintain the integrity and consistency of the SPL. The overall coordination must ensure that the products and core assets remain aligned with each other [1].

Products in an SPL share a reference architecture, which specifies the common structure of the products and centers in the development and evolution of both core assets and application products. Architectural SCM acknowledges the central role that the software architecture plays in software development and maintenance [2]. Therefore, architectural SCM is an effective means for evolution management in SPL. There have been some related works on evolution management for SPL architecture, e.g. the xADL-based works ([4][5][6]). These works focus on evolutions of architectures only and do not provide supports for component evolutions, evolution integration and product release, etc.

In this paper, we propose an architecture-based evolution management method for SPL. The method provides coordinated and unified evolution management for SPL, which can keep the continuous optimization of core assets, and at the same time support the implementation of customer requirements in each product. On the other hand, the method supports evolutions of both architecture/component specifications and component implementations. In the method, evolution managements for architectures and components are separated, and a comprehensive version model for both core assets and products is proposed to form the basis of evolution integration and release configuration.

The remainder of this paper is organized as follows. Section 2 introduces some related work and compares our works with them. Section 3 presents the evolution management method. Section 4 presents a case study and evaluates our method. Finally, we draw conclusions and discuss future work in section 5.

## 2. Related work

Traditional SCM tools are designed with the intention of versioning a single product, so they do not have facilities to support forward and backward change propagations [7]. Van Gurp et al. [8] propose to combine product derivation and variability management based on existing version management



tools (e.g. Subversion [9]). Yu et al. [3] propose an evolution-based SCM model for SPL, but no detailed introduction on architecture and component evolutions are reported. Thao et al. [7] present a SCM system MoSPL for product derivation in SPL. MoSPL provides version management at the component level, and explicitly manages logical constraints and derivation relations among components, thus enabling the automatic propagation of changes in core assets to products and vice versa. Their method concentrates on component-level derivation and evolution only.

The xADL group has a series of works on evolution management for SPL architecture. They present Ménage, the xADL 2.0 based environment for managing evolving SPL architectures in [6]. The tool provides supports for architectural element versioning and reference architecture customization. Their architecture differencing and merging method is presented in [5]. However, evolution synchronization and component evolutions are not mentioned.

Our method adopts xADL 2.0 to represent both reference and application architectures also. However, our evolution management method differs from theirs at several aspects: evolutions of both abstract architecture models and component implementation are supported; both evolutions of core assets and application products are involved with periodic synchronizations. Furthermore, in architecture merging, our method adopts the policy of variability abstract on the differences among the reference architecture and application architectures, not the all-included merging in [5].

### 3. Our method

#### 3.1 xADL 2.0

xADL 2.0 [4] is a highly-extensible, XML-based architecture description language, which includes a set of schemas to describe the architecture of a single software system or a product line. The most important part is the **Structure&Type** schema, which is used to describe basic architectural elements at design time, including components, connectors and links. Each component in architecture can have a component type describing the type information of the architectural components, including signatures, etc.

Architectural variability for SPL is supported by the **Options** and **Variants** schema. Options indicate points of variation in an architecture where the structure may vary by the inclusion or exclusion of a group of elements [4]. Variants indicate points in an architecture where one of several alternatives may be

substituted for an element or group of elements [4]. Each optional or variant element is accompanied by a guard condition to determine the inclusion or exclusion of it. Readers can refer to [4] for detailed introductions to xADL 2.0.

Figure 1 depicts an xADL-style reference architecture of the online book shopping product line. In the architecture, there are optional component *AcctMgtUI* and *AcctMgt* and variable component *Payment* and *Discount*. *Payment* has two variants of *PayByVirtualCur* and *PayByCreditCard* for two different modes of payment. *Discount* has no variants, implying that each application can have different discount policy, so it is an abstract component to be instantiated in application engineering. Guard conditions for these optional and variant components are also listed in Figure 1. It can be seen that payment mode (represented by the symbol *payMode*) is the main variation point. And variation constraints are implied by guard conditions: if setting *payMode* to be *virtualCur* then components for account management (*AcctMgtUI* and *AcctMgt*) should also be bound.

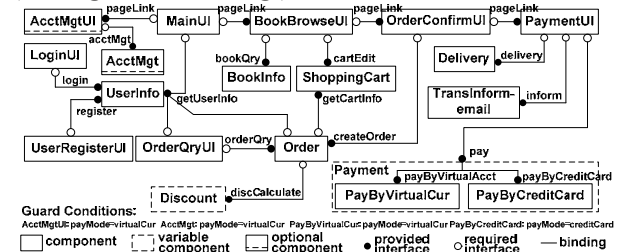


Figure 1. Reference architecture of the online book shopping product line

#### 3.2 SPL version model

The version model of our method extends the structure model of xADL 2.0 schemas of **Structure&Type**, **Options** and **Variants**. The model is depicted in Figure 2, in which grey boxes represent elements from xADL 2.0 and others are our extensions.

From the model, we can see that both architectures and components are versioned entities. In each product line, there is only one reference architecture (**RefArch**), and all the application architectures (**AppArch**) are derived from it. Both **RefArch** and **AppArch** can have multiple versions, and each version is composed of a set of **Component**, **Connector** and **Link**. Besides, there are architectural variations (optional and variant components) and variation constraints in each **RefArch** version. Each **AppArch** version may be synchronized with a **RefArch** version or not, representing the independent evolutions and periodic

synchronizations of **RefArch** and **AppArch**. By synchronization, we mean that the adaptations of an application asset are within the variability scope of corresponding domain asset. Similarity, there are both domain and application components, their versions and derivation/synchronization relationships between them.

In our method, component specifications are separated from component implementations as independent versioning entities. Each component implementation declares a component specification as its type and then each version of it will implement a specification version, representing that the component implementation complies with the specification. To distinguish the evolutions of architectures and components, we assume that each **ComponentType** in xADL 2.0 refers to a component specification version in our model, thus component implementations are completely separated from architectures.

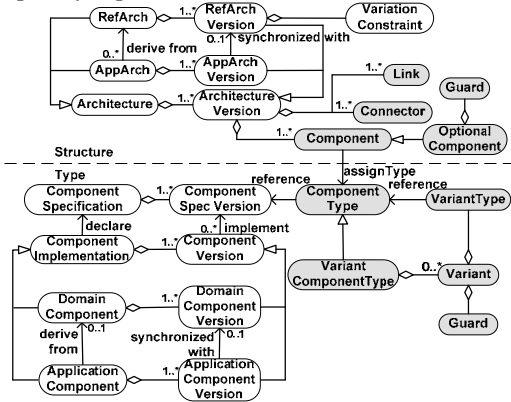


Figure 2. SPL version model

### 3.3 Evolution process

In our method, core assets and products can evolve independently. Temporary deviations are allowed, and periodic synchronizations on both architecture and component level will be performed to reunify core assets and application products.

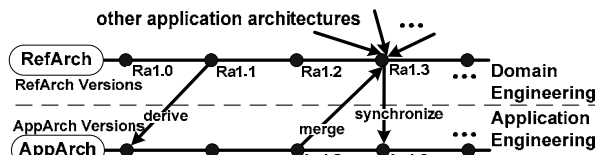


Figure 3. Architecture evolution process

Architecture evolution process in our method is presented in Figure 3. We can see that besides independent reference or application architecture evolutions, there are also cross evolution paths, including architecture derivation, architecture merging

and synchronization. The first version of an application architecture is always derived from the latest version of the reference architecture, e.g. **Aa1.0** is derived from **Ra1.1** and naturally they are synchronized. After that, the application architecture can evolve independently (e.g. **Aa1.1** and **Aa1.2**). On the other hand, reference architecture may also evolve for design optimization or new features (e.g. **Ra1.2**). After some time, periodic merging is performed among current versions of the reference architecture and all application architectures to make a new reference architecture version (e.g. **Ra1.3**). This merging propagates architecture evolutions in application products to the reference architecture. After that, synchronization is performed to propagate evolutions of reference architecture back to application architectures. Then, reference architecture and application architectures are synchronized again (e.g. **Ra1.3** and **Aa1.3**).

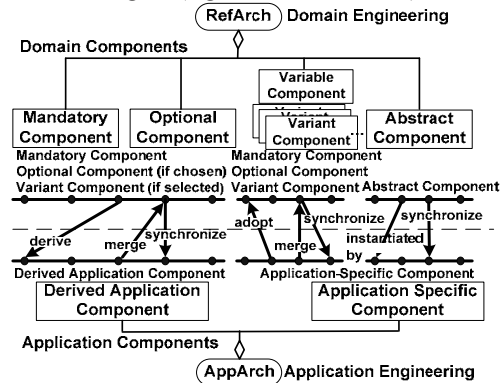


Figure 4. Component evolution process

In our architecture-centric evolution management, component evolutions are managed in term of their variability types, as shown in Figure 4. Derived application components are first derived from domain components along with the application architecture derivation or synchronization. It can be derived from a mandatory component, or an optional or variant component that is selected in architecture customization. After derivation, the application component can evolve independently and periodically be merged with corresponding domain component. Application-specific components are first created entirely for an application along with application architecture evolution. It may be a new part added to the architecture or a new variant for variable domain component. It will be evaluated in architecture merging by the domain architect and may be adopted as a domain component if the architectural extension is accepted into the reference architecture. Besides, an application-specific component can also be the

application-specific implementation (instantiation) for an abstract domain component. In this case, only synchronization on component specification should be considered in following evolutions, since abstract domain component specifies type information only.

### 3.4 Architecture evolutions

**3.4.1 Architecture derivation** Guard conditions in xADL 2.0 provide a built-in mechanism for automated application architecture derivation, e.g. the xADL

environment [6] provides the SELECTOR component. A guard condition is a Boolean expression composed of symbol, value and the comparison between the two parts, i.e. equal to, greater than, etc. A symbol can be used in several guard conditions for different optional or variant components. In architecture derivation, the application engineer will be requested to assign values to all the symbols, then all the optional or variant components can be determined to be bound or not according to the value of their guard conditions.

**Table 1. Merging policies for different kinds of architectural differences**

Difference Type	Description	Merging Policy
<i>NewComponent</i>	A <b>Component</b> in <b>AppArch</b> does not exist in <b>RefArch</b> , and the new <b>Component</b> links to at least one <b>Component</b> that corresponds to non-abstract <b>Component</b> in <b>RefArch</b>	merged as new mandatory <b>Component</b>
		merged as new optional domain <b>Component</b>
<i>NewComLinkToAbs</i>	A <b>Component</b> in <b>AppArch</b> does not exist in <b>RefArch</b> , and the new <b>Component</b> links to a <b>Component</b> that corresponds to an abstract <b>Component</b> in <b>RefArch</b>	N/A
<i>OptionalComRemoval</i>	Optional <b>Component</b> is removed in <b>AppArch</b>	N/A
<i>MandComRemoved</i>	Mandatory <b>Component</b> is removed in <b>AppArch</b>	change the mandatory <b>Component</b> to be optional
<i>DifComSpecVersion</i>	Non-abstract <b>Component</b> has the same component specification in <b>AppArch</b> , but with a new component specification version	merged with the domain component specification to make a new version
<i>NewComponentType</i>	Non-abstract <b>Component</b> has a new component specification in <b>AppArch</b>	merged as alternative <b>Component</b> and replace the original mandatory <b>Component</b>
<i>VariantBound</i>	Variable domain <b>Component</b> is customized to one of its prescribed variant in <b>AppArch</b>	N/A
<i>DifVariantComSpecVersion</i>	Variable domain <b>Component</b> is customized to one of its prescribed variant in <b>AppArch</b> , and the variant has a revised specification (new component specification version)	merged with the variant component specification to make a new version
<i>NewVariantComType</i>	Variable domain <b>Component</b> is customized to a new added variant (with new component specification) in <b>AppArch</b>	Add the new application variant to the variable domain <b>Component</b>
<i>AbsComInstance</i>	Abstract <b>Component</b> is replaced by an application-specific component in <b>AppArch</b>	N/A

**3.4.2 Architecture merging** Architecture merging in our method is performed on architectural differences between reference architecture and application architecture, which can be captured by our SPL evolution management environment. The differences can occur at the structure or type level. For example, removing a component is a structural difference, and replacing a variable component with a new variant is a type difference.

We identify 10 kinds of basic architectural differences, including architectural customizations as listed in Table 1, in which type differences are represented by grey lines. *VariantBound*, *OptionalComRemoval*, *AbsComInstance*, *NewComLinkToAbs* are architectural customizations within prescribed scope, so no merging operations are needed. Other cases are differences beyond the variability scope and merging operations will be performed. For example, in the cases of both *NewComponent* and *NewComLinkToAbs*, a new component is added in the application architecture. In

*NewComLinkToAbs*, the component is linked to an application component corresponding to an abstract component in the reference architecture, so it is considered to be part of the instantiation for the abstract component. In *NewComponent*, the component is linked to non-abstract domain components, so it is considered to be an additional architecture adaptation, e.g. the application engineer may decide to add an logging component to the *Order* component shown in Figure 1 for better security.

Merging policies for those architectural differences are listed in Table 1. In our method, differences on component specification mean completely different components (e.g. new variant), while differences on component specification version mean revised component specifications (e.g. adding an interface or interface revisions). It can be seen that in some cases user intervention is needed, e.g. to determine whether merged as mandatory or optional component in *NewComponent*. After merging, new architectural variation points or functional extensions may be added

to the reference architecture, e.g. adjusting mandatory components to be optional, or accepting new domain components from application architectures.

**3.4.3 Architecture synchronization** After architecture merging, the reference architecture embodies all the application differences by new variation points, which makes it possible to synchronize application architectures. Architecture synchronization is to propagate evolutions in reference architecture, from both itself and other applications, to each application architecture. It can be seen as the re-derivation of application architecture from the new reference architecture version.

The symbols in xADL 2.0 represent business or design options independent of specific variation points, so the customization decisions (symbol value assignments) can be reused. For those newly added symbols, the application engineer will be requested to assign values for them.

### 3.5 Component and product evolution

Component-level evolutions include individual component evolution and cross evolution also. Individual component evolution may be due to revision of specification or implementation only. For a component, evolution may be due to new specification it implements (e.g. adding a new interface) or purely an implementation revision (e.g. bug fixing). The former is supported by versioning of component specifications and the management of the implementation relations between component implementations and specifications (see Figure 2). The latter is implemented by file-level evolution management and can be supported by traditional version management system, e.g. Subversion [9] integrated in our implementation.

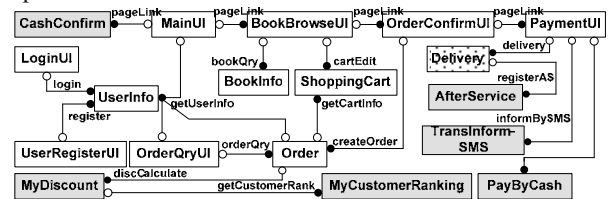
Among those cross component evolutions depicted in Figure 4, component merging and synchronization are the main problems. In component merging, derived application component versions will be merged into corresponding domain component. It is file-based merging of component implementations, so the merging can be supported by the version merging mechanism in traditional SCM systems. Component synchronization is to propagate new version of a domain component to those application components derived from it. After synchronization, a copy of the domain component will become the current version of the application component.

Product evolution is supported by the product release mechanism. As mentioned before, evolutions on the architectural level and component level are separated in our method. Product release should first choose an application architecture version and then determine versions for all the components involved.

## 4. Case study and evaluation

Our method has been implemented in the prototype evolution management tool ASCMPL (Architecture-based Software Configuration Management tool for Product Line). It is an eclipse plug-in developed on Subversion [9] and xADL [4] library. ASCMPL provides direct support for architecture and component specification development. File-based evolutions of component implementations are managed by Subversion, and file-level versioning information (e.g. URLs and revisions in Subversion) is referred in component configuration information for integration.

In order to evaluate our method, we conduct a case study on an enterprise product line, i.e. the online book shopping system, with ASCMPL. It is a Java-based web system. Figure 1 shows the initial reference architecture of the product line, in which payment mode is considered as the main variation point (see the symbol *payMode* in the guard conditions). Based on this reference architecture, an application engineer derives a product variant and adapts the application architecture to meet application-specific requirements. The adapted application architecture is shown in Figure 5, in which grey blocks represent new application components and dotted blocks represent components with modified specifications.



**Figure 5. Adapted application architecture**

According to our architecture merging method, we identify 9 architectural differences of 7 different types and corresponding merging operations as shown in Table 2. After merging, new variation points are introduced, including optional component *CashConfirm*, *AfterService* and alternative component *TransInform*. Besides, two new symbols of *afterService* and *inform* are added and a new candidate value *cash* is added for the existing symbol *payMode*. Due to the limitation of space, the reference architecture after merging is not presented. In this case, some component-level evolutions are also involved, e.g. component-level merging between domain component *Delivery* and the modified application component *Delivery*. In following evolution synchronizations, these new features in the reference architecture will be propagated to other applications and their existing customization decisions can be reused. For example, an

application with the component *PayByCreditCard* can reuse the decision “*payMode=creditCard*” in the synchronization, and decisions for new symbols (e.g. *afterService*) should be complemented of course.

From the case study, it can be seen that for a real software product line, long-term and coordinated evolution management is necessary. Architectural SCM is essential for SPL evolutions, since both of them acknowledge the central role of architecture. Moreover, in order to provide comprehensive evolution management, both specification- and implementation-level evolutions should be supported. Our evolution management method provides an integrated version model for both architectures and components. Based on the version model, the method supports both forward architecture derivation and backward evolution feedbacks by architecture merging and synchronization. It also provides the mechanism to integrate traditional file-based SCM tools to make a comprehensive evolution management for SPL.

**Table 2. Architecture differences and merging operations in the case study**

Difference	Difference Type	Merging operations
<i>PayByCash</i>	<i>NewVariantComType</i>	as a new variant of <i>Payment</i> with guard condition “ <i>payMode=cash</i> ”
<i>CashConfirm</i>	<i>NewComponent</i>	as a new optional component with guard condition “ <i>payMode=cash</i> ”
<i>AcctMgt</i>	<i>OptionalComRemoval</i>	N/A
<i>AcctMgtUI</i>	<i>OptionalComRemoval</i>	N/A
<i>MyDiscount</i>	<i>AbsComInstance</i>	N/A
<i>MyCustomerRanking</i>	<i>NewComLinkToAbs</i>	N/A
<i>Delivery</i>	<i>DifComSpecVersion</i>	merged with the domain component to produce a new specification version
<i>AfterService</i>	<i>NewComponent</i>	as a new optional component with guard condition “ <i>afterService=true</i> ”
<i>TransInform-SMS</i>	<i>NewComponentType</i>	merged with <i>TransInform-email</i> to make a new alternative component <i>TransInform</i> with guard condition “ <i>inform=SMS</i> ” and “ <i>inform=email</i> ”

## 5. Conclusion and future work

In this paper, we present an architecture-based evolution management method for SPL. A version model involving both domain and application architectures and components is proposed and evolution

processes for architectures and components are presented. The method supports architecture merging and corresponding evolution synchronization. For component-level evolutions, our method supports both specification and implementation evolution.

In our future work, we will try to integrate feature model [10] and other SPL artifacts in the evolution management on certain feature-based traceability mechanism. On the other hand, we will try to integrate the evolution management with our product derivation tool [11] to provide a complete platform for incremental SPL development.

**Acknowledgments.** This work is supported by National Natural Science Foundation of China under Grant No. 60703092, and National High Technology Development 863 Program of China under Grant No. 2007AA01Z125.

## References

- [1] P. C. Clements, L. G. Jones, L. M. Northrop, J. D. McGregor. Project Management in a Software Product Line Organization. IEEE Software, 2005, 22(5).
- [2] B. Westfechtel, R. Conradi. Software Architecture and Software Configuration Management. In SCM 10, 2001.
- [3] L. Yu and S. Ramaswamy. A Configuration Management Model for Software Product Line. INFOCOMP Journal of Computer Science, 2006, 5 (4).
- [4] E. M. Dashofy, A. Hoek, R. N. Taylor. A Comprehensive Approach for the Development of Modular Software Architecture Description Languages. TOSEM, 2005, 14 (2).
- [5] P. Chen, M. Critchlow, A. Garg, et al.. Differencing and Merging within an Evolving Product Line Architecture. In PFE’03, 2003.
- [6] A. Garg, M. Critchlow, P. Chen, et al.. An Environment for Managing Evolving Product Line Architectures. In ICSM’03, 2003.
- [7] C. Thao, E. V. Munson, T. N. Nguyen. Software Configuration Management for Product Derivation in Software Product Families. In ECBS’08, 2008.
- [8] J. Gurf and C. Prehofer. Version Management Tools as a Basis for Integrating Product Derivation and Software Product Families. Variability Mgmt., Workshop at SPLC’06.
- [9] Subversion. <http://subversion.tigris.org>.
- [10] X. Peng, W. Zhao, Y. Xue, Y. Wu. Ontology-Based Feature Modeling and Application-Oriented Tailoring. In ICSR’06, 2006.
- [11] X. Peng, L. Shen, W. Zhao. Feature Implementation Modeling based Product Derivation in Software Product Line. In ICSR’08, 2008.

# Towards design and architectural evaluation of product variants: A case study on an open source software system

Muhammad Irfan Ullah<sup>†</sup>, Guenther Ruhe<sup>†‡</sup>, Vahid Garousi<sup>‡</sup>

<sup>†</sup>Department of Computer Science, <sup>‡</sup>Department of Electrical and Computer Engineering,  
University of Calgary, Canada  
{miullah, ruhe, vgarousi}@ucalgary.ca

## Abstract

*Evolving a software system demands a careful balance between equally important but often conflicting views of customers and system architecture. This paper proposes a method to address evolution of a software system into a product line containing specialized product variants for specific markets while aligning the two views. The proposed method COPE+ iteratively explores the solutions space to generate product variants for the two views independently. It uses density based clustering to identify market segments. Impact of the proposed features on the existing product's architecture is heuristically determined. Behaviors of the promising variants are then compared with that of the existing system through extended mq-simulation on statechart representations. This determines the degree of similarity between existing system and proposed product variants. Finally, human experts evaluate the suggested products. COPE+ is applied to jEdit, a popular open source editor. Results indicate usefulness of the proposed method in bringing together the diversified views of customers and architecture.*

## 1. Introduction

A software product line (SPL) is a set of software intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission [1]. SPL is a viable approach if a company plans to target a wide and diverse customer base. Typically SPLs evolve from existing products or systems that are successful and therefore attract customers from a wide variety of domains. A number of real-world case studies show the presence of this phenomenon, e.g., CelsiusTech Ship Systems 2000 and Cummins Inc. diesel engine SPL to name a few [1]. As a note on terminology, we use the term product variant to refer to an individual

product in a SPL. Helferich et al. [2] report that in the product definition phase many of the existing SPL development methodologies either focus on technical details [3] without involving customers in this process or they identify marketing techniques for this purpose without prescribing how to translate the results of these techniques into tangible products [1]. Proposed method COPE+ attempts to address this shortcoming for the specific evolution scenario when an existing (single) software system is evolved into a product line. It builds upon and improves our previous work (COPE) [4] as following:

1. Feature impact analysis on existing system architecture using heuristics.
2. Evaluation of customers' proposed product variants using results of impact analysis in 1.
3. Behavioral comparison of selected product variants with existing system using statecharts.

The remainder of this paper is organized in seven sections. Section 2 presents the problem statement. Section 3 introduces technical concepts included in COPE+. Section 4 presents related work. Section 5 presents proposed method COPE+. Section 6 illustrates the method using jEdit system. Section 7 discusses applicability and value of COPE+ and Section 8 presents future work.

## 2. Problem Statement

This paper addresses the question: "How can an existing software system facing feature requests from a diverse customer-base be evolved into a product line with product variants targeting different market segments such that the impact on system architecture is reduced?" Cost-benefit analysis of impact on architecture has not been done in this work.

## 3. Background

In this section we introduce two technical concepts that will be used in COPE+.

### 3.1 Statecharts

We use statechart (also referred to as state transition diagram) [5] representation of the system to evaluate behavioral similarity between an existing system and its product variants. Statechart representation makes it possible to perform this comparison in an operational and systematic manner. A statechart is a directed graph  $G = (S, \mathcal{L}, T, s_0)$  where

$S$  is a set of states

$\mathcal{L}$  is a set of labels of transitions

$T$  is a transition relation such that  $T \subseteq S \times \mathcal{L} \times S$

$s_0 \in S$  is the initial state

### 3.2 Simulation-based Comparison of Statecharts

Structure-based (*cost and feature-based*) similarity measures for statecharts are not very useful where semantic information is important [6]. Therefore, we selected a behavioral based comparison method: *extremal quantitative simulation* (mq-simulation, for short) [7] which considers semantics of the statecharts while calculating similarity. Below, we briefly explain simulation based comparison of statecharts.

*Simulation:* If a statechart  $G^1$  has all of the behaviors of a statechart  $G$  and maybe more then  $G^1$  completely simulates  $G$ . Similar simulation relation can be established from  $G$  to  $G^1$  [7].

*Bisimulation:* It is a two way simulation.  $G^1$  and  $G$  bisimulates each other if  $G^1$  simulates  $G$  and  $G$  simulates  $G^1$ .

However, mq-simulation in its original form is not applicable to our problem since it does not allow partial similarity. Additionally, for product line design it is important to identify variation points in product variants with respect to existing system, hence we extended mq-simulation to address these issues.

For  $G^1 = (S^1, \mathcal{L}^1, T^1, t_0)$  simulates  $G = (S, \mathcal{L}, T, s_0)$  we extend mq-simulation as following:

1. Label matching between any two transitions  $a$  and  $b$  can also accommodate partial similarity. Note that originally the results were binary [0, 1].

$L(a, b): \mathcal{L} \times \mathcal{L} \rightarrow [0 .. 1]$

2. A set of variation points initialized as  $VP = \{\}$  will be maintained during the simulation process and updated with respective states where mismatch occurs.

$G^1$  simulates  $G$  results in a measure  $Q(s_0, t_0)$  with value in the range [0 .. 1] and a set  $VP_a$

$G$  simulates  $G^1$  results in a measure  $Q(t_0, s_0)$  with value in the range [0 .. 1] and a set  $VP_b$

We combine the two simulation results to evaluate the bisimulation between  $G^1$  and  $G$  as:

$$G^1 \text{ bisimulates } G = \frac{Q(s_0, t_0) + Q(t_0, s_0)}{2}$$

Average is just one of the ways to combine the two simulation results. We think it is reasonable to calculate a mean value to compare bisimulation of various proposed product variants. However, other measures such as addition of two simulation results for each product variant can also be done.

$$VP = VP_a \cup VP_b$$

The rationale of performing union on the two sets is to get variation in both directions of simulation. Details of our extended mq-simulation model and its application can be found in [8].

## 4. Related Work

As mentioned in Section 1, COPE+ bridges the gap between customers and system architecture which has been declared as a major deficit in software product line development methodologies [2]. Scoping is one of the most critical activities in early phases of SPL development, however, it typically generates results based on economic and technical considerations and does not include customers' input in product definition [3]. Other related works to COPE+ are FAAM [9] and QFD-PPP [10], however, these methods have an implicit assumption of greenfield development and there is no consideration for existing product's architecture. Feature Oriented Domain Analysis (FODA) [11] method of SEI investigates product features to define domain for a set of related products. Like COPE+, it maps product features on architectural components and uses statecharts for behavioral representation of the system. However, FODA's goal (domain definition) is different from that of COPE+ (product evolution).

Kuhn et al [12] have proposed the idea of semantic clustering for refactoring of software systems. We have used this concept for identifying the impact of proposed features on existing system architecture. For behavioral comparison of systems, COPE+ extends the work of Sokolsky et al [7].

## 5. Method: COPE+

COPE+ is a decision support method architected on the concept of Hybrid Intelligence as implemented in EVOLVE\* [13] for the problem of release planning. The idea of decision support systems is to suggest the most qualified solution(s) from a large solutions space using advanced computational techniques to the human expert who can select the one that is most promising using soft and implicit objectives. The rationale behind the concept of hybrid approaches is that the solution generated by combining human and

computational intelligence is better than the one generated by applying them in isolation [13]. COPE+ has three phases as shown in Figure 1. The focus of this paper is on Exploration phase where we introduce computational techniques to systematically evaluate customers' suggested products with existing architecture.

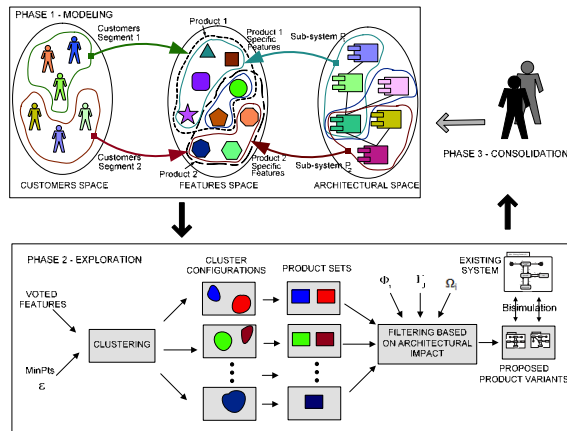


Figure 1: An overview of the iterative method COPE+

### 5.1 Phase 1 - Modeling

The first phase of COPE+ formalizes the problem in three domains, i.e., customers (left oval in fig. 1, Phase 1), architecture (right oval in fig. 1, Phase 1) and proposed product features (middle oval in fig. 1, Phase 1). Relationships are established amongst these domains through customers' voting on the features and impact analysis of features on architecture.

Customers are represented by  $C = \{c_1, c_2, \dots, c_i\}$ . For simplicity, it is assumed that all the customers have equal importance. Features are represented as  $F = \{f_1, f_2, \dots, f_m\}$ . The granularity of a feature is not preset. It can be a single feature request or multiple feature requests grouped together on any given criterion. Customers vote on the features based on a 9-point likert scale (1: least desired to 9: extremely desired). A vote represents *value* of the feature to a customer. Additional criteria can also be defined for voting.

Existing system architecture is evaluated at the abstraction level of packages (a package is a composition of classes) which are represented as  $K = \{k_1, k_2, \dots, k_n\}$ . Each one of the proposed features  $f_i$  impacts a set of packages  $\Phi_i \subseteq K$ . Such  $\Phi_i$  can be generated through any impact analysis technique such as [12]. We refer to  $\Phi_i$  as *Package Impact Set* for feature  $f_i$ . In the opposite direction, each package  $k_j$  (partially) implements a set of features  $\Gamma_j \subseteq F$  referred to as *Feature Impact Set* for package  $k_j$ . A dependency relationship is established amongst the packages implementing the same feature. This relationship is defined by an  $n \times n$  matrix  $\Psi$  such that for any two packages  $x$  and  $y$ , if  $\Phi_x \cap \Phi_y \neq \{\}$  then

$\Psi(x,y) = 1$  otherwise  $\Psi(x,y) = 0$ . Sum of all the entries in column (or row)  $j$  gives the *Dependency Value*  $\Omega_j$  for the package  $j$ .

### 5.2 Phase 2 - Exploration

This phase gets inputs from the Modeling phase and uses them to explore the solutions space. On the customers' side, DBSCAN [14] clustering algorithm is applied using RapidMiner version 4 to customers' voting on proposed features. Two additional input parameters,  $\epsilon$  (neighborhood distance) and MinPts (minimum number of data points in a cluster) required by DBSCAN are also evaluated. Interested reader is referred to [4] for details on the reasons for selection of DBSCAN and its application. By varying the value of  $\epsilon$ , all possible cluster configurations  $O_a = \{o_{a1}, \dots, o_{aq}\}$  are generated where as each  $o_{ai}$  is a customers cluster representing a market segment. A products set  $P_a = \{p_{a1}, \dots, p_{aq}\}$  for each cluster configuration  $O_a$  is proposed containing products  $p_{ai}$  corresponding to market segments  $o_{ai}$ .

On the architecture side, the impact of implementing the proposed features is calculated. Three heuristics based on greedy algorithms are used independently to select features for implementation such that the impact on existing architecture is reduced. H1 is illustrated in Algorithm 1, given below. H2 and H3 are defined similarly [8].

H1: Package with smallest *Dependency Value* first.

H2: Package with least number of classes first.

H3: Package with least number of lines of code first.

#### Algorithm 1: ARCHITECTURAL\_IMPACT

**Inputs:** For each feature  $f_i$ , Package Impact Set  $\Phi_i$ . The set of all packages  $K$ . For each package  $k_j$ , Dependency Value  $\Omega_j$  and Feature Impact Set  $\Gamma_j$ .

**Output:** A sequence of implementation for features  $f_i$  such that total number of packages impacted is minimized.

```

1. BEGIN
2. BinA = K, BinB = {}, QueueC = {}
3. WHILE BinA is NotEmpty
4.   BinB = Package  $k_j$  from BinA with smallest  $\Omega_j$ 
5.   BinA = BinA -  $k_j$ 
6.   Search  $\Gamma_j$  for  $f_i$  such that  $|\Phi_i|$  is smallest
7.   BinB =  $\Phi_i$ 
8.   BinA = BinA -  $\Phi_i$ 
9.   QueueC =  $f_i$ 
10. END WHILE
11. RETURN QueueC
12. END

```

Each heuristic identifies a set  $U_{Hi}$  of *cut-points*. A *cut-point* is defined as a group of features formed by combining adjacent iterations when new packages added from iteration  $i$  to  $i+1$  are less than the *threshold*  $t$ . The value of *threshold* is selected based on the data set under consideration, additionally; a range of threshold values can also be used. As features are added to the group within a *cut-point* boundary, very few new packages are impacted but as we move across to the next *cut-point*, a large number of packages are impacted even for including one new feature. Product sets  $PS_a$  proposed by the



customers are then evaluated using these *cut-points*. Most promising product set is selected for further evaluation. Up to this point all the analysis on the products was static. The behaviors of the product variants in the selected product set are now analyzed in more detail by comparing them with the existing system. We use statechart representation of the system to evaluate behavioral similarity through bisimulation between each proposed product variant and the existing product as presented in Section 3.2. The sequence of activities in this phase is shown in Figure 1, Phase 2: Exploration.

### 5.3 Phase 3 - Consolidation

In the third phase, human experts analyze the results for proposed product variants. They are able to address also tacit concerns not being handled in the formalized solution method in Phase 2. As a result, the experts can also identify certain changes to the underlying model to generate more appropriate solutions in the next iteration. The detailed discussion of Consolidation phase is out of scope of this paper and will be presented in follow up work.

## 6. Application of COPE+ on jEdit

We have applied COPE+ to jEdit v4.0 (www.jedit.org) which is a popular open source text editor.

### 6.1 Phase 1 - Modeling

A total of ten customers belonging to diverse domains were hypothetically asked to vote on the proposed feature groups i.e.  $l = 10$ . We selected 95 feature requests from the jEdit project website [15] while 14 feature requests were hypothetically created. To determine the impact of these features on the existing system, we classified them into nine groups based on their functionality, using results of [12]. All the analysis in this case study has been performed at the level of feature groups as shown in Table 1, therefore,  $m = 9$ . jEdit v4.0 has thirty packages containing 394 classes. Impact analysis is performed at the level of packages hence,  $n = 30$ . Evaluation of *Package Impact Set*  $\Phi_i$ , *Feature Impact Set*  $\Gamma_j$  is partially shown in Table 1.

Table 1: jEdit feature groups and architectural impact

ID	Feature Group	Functionality	Packages			asm ( $\Gamma_{30}$ )
			util ( $\Gamma_1$ )	options ( $\Gamma_2$ )	browser ( $\Gamma_3$ )	
1	DC ( $\Phi_1$ )	Domain Concepts	X	X	X	
2	UI ( $\Phi_2$ )	User Interface		X	X	
3	RE ( $\Phi_3$ )	Regular Expressions	X			
4	TB ( $\Phi_4$ )	Text Buffers		X		
5	DW ( $\Phi_5$ )	Dockable Windows				
6	BS ( $\Phi_6$ )	Beanshell Scripting				
7	XR ( $\Phi_7$ )	XML Reader				
8	BA ( $\Phi_8$ )	Bytecode Assembler				X
9	TZ ( $\Phi_9$ )	Tar and Zip Archives	X			X

A cross in a cell (Table 1) means feature group  $f_i$  impacts package  $k_j$  and conversely package  $k_j$  (partially) implements feature group  $f_i$ . Collecting all entries in row  $i$  forms the set  $\Phi_i$ , doing the same for column  $j$  results in  $\Gamma_j$ . *Dependency Value*  $\Omega_j$  for each package of jEdit is evaluated through an  $n \times n$  packages interaction matrix referred to as  $\Psi$ . Detailed data and analysis of this example are presented in [8].

### 6.2 Phase 2 - Exploration

We have used a range of values for  $\epsilon$  (1 to 18) to generate all possible customers' cluster configurations using DBSCAN [14]. Results for three such configurations are shown in Table 2. Last column shows market segments containing customers.

Table 2: Cluster configurations using DBSCAN

$\epsilon$	Configuration	No. of Clusters	Market Segments
12	$O_1$	2	$o_{11}$ : 1, 3, 9, 10 $o_{12}$ : 5, 6, 7, 8
15	$O_2$	2	$o_{21}$ : 1, 3, 4, 9, 10 $o_{22}$ : 2, 5, 6, 7, 8
18	$O_3$	1	$o_{31}$ : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Product sets for these cluster configurations are shown in Table 3. Product variants within each product set are generated by combining the features highly desired (e.g., voted 7 or higher on a 9-point likert scale) by the customers in corresponding market segment.

Table 3: Products sets for market segments

Products Set	Product Variants
PS <sub>1</sub>	$p_{11}$ : XR, UI, DW, TB $p_{12}$ : UI, TB, BS, DC, TZ
PS <sub>2</sub>	$p_{21}$ : XR, UI, DW, BS $p_{22}$ : UI, TB, RE, DC
PS <sub>3</sub>	$p_{31}$ : XR, UI, DW, TB, RE, DC

Application of the heuristics H1, H2 are shown in Figures 2 and 3 respectively. Illustration of H3 is not shown due to space shortage. These impact diagrams identify the *cut-points* as dashed vertical lines using threshold  $t=2$ , which is reasonable for the given data. However, other threshold values can also be used to generate multiple sets of results for each heuristic.

Table 4 summarizes the impact of implementing proposed features on the jEdit architecture. For each heuristic, the set of cut-points is presented. The values in the last column (architectural impact) are calculated using the impact diagrams for corresponding heuristics (e.g., Figures 2 and 3).

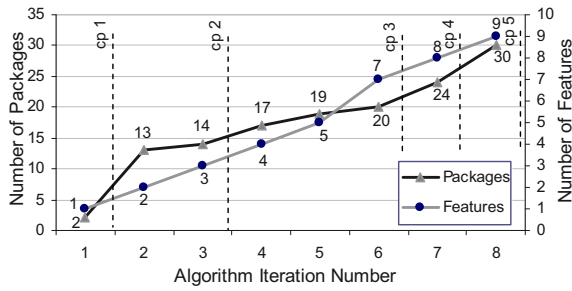


Figure 2: Impact on jEdit's architecture using H1

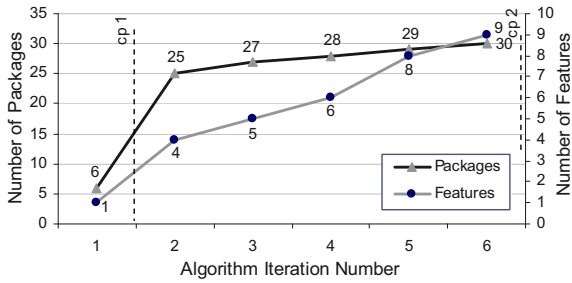


Figure 3: Impact on jEdit's architecture using H2

Now we evaluate the customers' proposed products in Table 3 with architectural impact as shown in Table 4. Results are shown in Table 5.

Table 4: Feature impact on jEdit's architecture

Heuristic	Cut-point Set	Features	Architectural Impact
H1	$U_{H1} = \{cp1, cp2, cp3, cp4, cp5\}$	cp1 = {XR} cp2 = {UI, DW} cp3 = {TB, RE, TZ, BA} cp4 = {BS} cp5 = {DC}	cp1 = 6.7% cp2 = 46.7% cp3 = 66.7% cp4 = 80% cp5 = 100%
H2	$U_{H2} = \{cp1, cp2\}$	cp1 = {BS} cp2 = {DC, UI, DW, TB, XR, TZ, BA, RE}	cp1 = 20% cp2 = 100%
H3	$U_{H2} = \{cp1, cp2\}$	cp1 = {BS} cp2 = {DC, UI, DW, TB, XR, TZ, BA, RE}	cp1 = 20% cp2 = 100%

Table 5 shows that all product sets perform equally using H2 and H3. However, PS1 has the least impact on the architecture using H1. Hence it is selected for behavioral comparison with the existing system.

Table 5: Impact of product variants on jEdit's architecture

Product Set	Product Variant	Arch. Impact (H1)	Arch. Impact (H2)	Arch. Impact (H3)
PS1	p11	66.7%	100%	100%
	p12	100%	100%	100%
PS2	p21	80%	100%	100%
	p22	100%	100%	100%
PS3	p31	100%	100%	100%

In this example, G (Figure 4) is the statechart representing simplified subset of behavior of the existing jEdit system. Performing analysis on a subset of behavior is valid because only the states

where product variants extend the existing system need to be examined. States are shown as rounded rectangles while arrows represent transitions with label  $a_i$  as events triggering those transitions. Due to space shortage, one event ( $a_1 = \text{open file}$ ) is shown as an example. We will compare the behavior of each proposed product variant of PS1 with the existing jEdit system through bisimulation. The statechart representations are generated by analyzing the design documentation of the existing system.

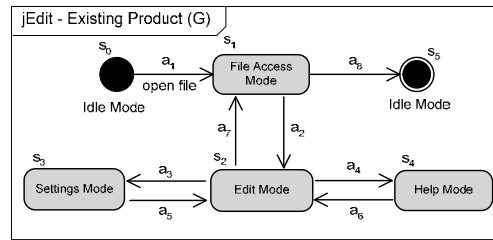


Figure 4: Statechart representing a simplified subset of jEdit's behavior

### Bisimulating Product Variant p11

A major new functionality proposed for product p11 is related to different types of user interfaces including tablets, table top displays for distributed team development [8]. Figure 5 represents behavior of p11 as statechart  $G^1$  including this additional functionality.  $G^1 = (S^1, \mathcal{L}^1, T^1, t_0)$  simulates  $G = (S, \mathcal{L}, T, s_0)$  results in  $Q(s_0, t_0) = 1$  which implies p11 has all the behavior of existing jEdit system. The set of variation points  $VP1_a = \{\}$ . For the opposite case i.e. G simulates  $G^1$  results in  $Q(t_0, s_0) = 0.66$  which means the existing jEdit system simulates the behavior of p11 up to 66%. This is because of the presence of new events  $b_9$  and  $b_{10}$  as well as a new state  $t_6$ . The variation points set  $VP1_b = \{s_2\}$  meaning  $s_2$  state in the existing jEdit system becomes a variation point in p11 initiating new behavior. The two results are combined to generate the bisimulation as following:

$$G^1 \text{ bisimulates } G = \frac{Q(s_0, t_0) + Q(t_0, s_0)}{2} = \frac{1 + 0.66}{2} = 0.83$$

$$VP1 = VP1_a \cup VP1_b = \{\} \cup \{s_2\} = \{s_2\}$$

### Bisimulating Product Variant p12

Customers for product p12 have requested inclusion of multiple user types with different access privileges to the project. Product p12 requires ability to define a team structure with approval hierarchy [8]. This translates to the statechart  $G^2$  in Figure 6 representing behavior of p12. The results of applying bisimulation between  $G^2$  and G are as following:

$$G^2 \text{ bisimulates } G = \frac{Q(s_0, t_0) + Q(t_0, s_0)}{2} = \frac{0.75 + 0.75}{2} = 0.75$$

$$VP1 = VP2_a \cup VP2_b = \{s_0, s_2\} \cup \{s_0, s_2\} = \{s_0, s_2\}$$

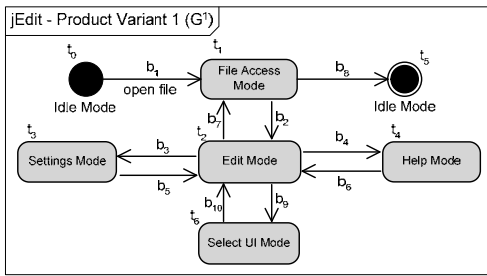


Figure 5: Statechart representing a simplified subset of p11's behavior

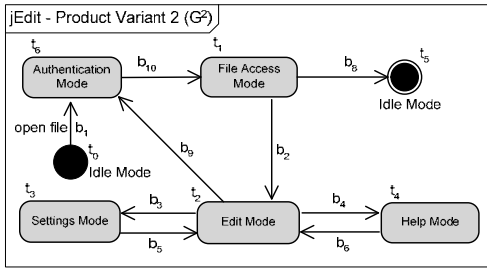


Figure 6: Statechart representing a simplified subset of p12's behavior

### 6.3 Phase 3 - Consolidation

In the final phase of COPE+ a human expert will evaluate the results of Exploration phase and evaluate the suggested products with respect to their bisimulation results. A trade off analysis can be performed between the bisimulation values and variation point sets to make product selection decision. Details are out of scope of this paper and will be presented in follow-up works.

## 7. Discussion

Evolving a software system is a complex problem. This complexity exponentially increases when there are multiple market segments requiring customized product variants. COPE+ helps decision makers by exploring the vast solutions space and selecting promising solutions for in depth evaluation. We have selected three different heuristics so as to identify an evolution strategy with the least impact on the system architecture. However, this is an ongoing effort and results presented in this paper have not been validated. Hence, we cannot confirm how much the architectural impact will be reduced by using these heuristics. Acceptance of a particular solution also requires proactive participation of customers for making trade-offs between architectural impact and feature selection.

## 8. Future Work

We plan to include more heuristics for architectural impact evaluation and eventually use evolutionary algorithms to generate more robust solutions. Further

case studies are planned to validate effectiveness, or otherwise, of COPE+ in aiding product evolution decisions. Cost-benefit analysis of proposed method will also be performed as part of these case studies including investigation of a convenient level to which impact on architecture should be reduced for acceptable results.

## Acknowledgements

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC Discovery Grant no. 250343-07). Vahid Garousi is supported by the NSERC Discovery Grant no. 341511-07 and also by the Alberta Ingenuity New Faculty Award no. 200600673.

## References

- [1] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley, August 2001.
- [2] A. Helferich, K. Schmid and G. Herzworm, "Product management for software product lines: An unsolved problem?", *Communications of the ACM*, Vol. 49, No. 12, December 2006, pp. 66-67
- [3] K. Schmid, "A comprehensive product line scoping approach and its validation," in proceedings of the *ICSE 2002*, Orlando, USA, pp. 593 – 603.
- [4] M. Ullah and G. Ruhe, "One product versus product line: Decision support based on customer needs analysis", *Doctoral Symposium at SPLC 2007*, Kyoto, Japan, pp. 174-183
- [5] J. Rumbaugh, I. Jacobson and G. Booch, *The Unified Modeling Language Reference Manual*, 2<sup>nd</sup> Ed., Pearson Education, July 2004.
- [6] A. Sanfeliu and K. Fu, "A distance measure between attributed relational graphs for pattern recognition", *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 13(3), pp. 353-362, 1983
- [7] O. Sokolsky, S. Kannan, and I. Lee, "Simulation-based graph similarity", in *TACAS*, pp. 426-440, 2006
- [8] M. Ullah, "COPE+ and its application", *Technical Report SERG-2009-01*, University of Calgary, Canada.
- [9] T. Dolan, *Architecture Assessment of Information-System Families: A Practical Perspective*, Ph.D. Thesis, Technical University Eindhoven, 2001
- [10] A. Helferich, G. Herzworm, and S. Schockert, "QFD-PPP: Product Line Portfolio Planning Using Quality Function Deployment", in proceedings of *SPLC 2005*, Rennes, France
- [11] K. Kang et al., "Feature-Oriented Domain Analysis (FODA) Feasibility Study", *Technical Report CMU/SEI-90-TR-021*
- [12] A. Kuhn, S. Ducasse and T. Girba, "Semantic clustering: Identifying topics in source code", *Information and Software Technology*, Vol. 49(2007), pp. 230-243
- [13] G. Ruhe, and A. Ngo-The, "Hybrid Intelligence in Software Release Planning", *International Journal of Hybrid Intelligent Systems*, Vol. 1(2004), pp. 99-110
- [14] M. Ester, H. P. Kriegel, H. Sander and X. Xu, "A Density Based Algorithm for Discovering Clusters in Large Spatial Data Sets with Noise", *KDD 1996*, Portland, USA
- [15] <http://sourceforge.net/projects/jedit/>

# Decision Support System Environment for Software Architecture Style Selection (DESAS v1.0)

Shahrouz Moaven<sup>1</sup>, Hamed Ahmadi<sup>2</sup>, Jafar Habibi<sup>1</sup>, Ali Kamandi<sup>1</sup>

<sup>1</sup>Sharif University of Technology, PO Box 11365-9517, Tehran, Iran

<sup>2</sup>Young Researchers Club, PO Box 14737-33985, Tehran, Iran

moaven@ce.sharif.edu, ha.ahmadi@qazviniau.ac.ir, jhabibi@sharif.edu, kamandi@ce.sharif.edu

**Abstract.** *In software systems development lifecycle making use of software architecture, especially by taking advantage of architecture styles and patterns, is an essential part which increases product's quality. Nowadays, in order to cover complexity of systems, combination of different architecture styles should be used; therefore ambiguous behaviors might occur. Hence, due to the critical need for toolsets capable of selecting suitable styles and patterns, an environment is proposed in this paper which can perfectly cover different aspects of the implementation of decision support system (DSS). The designed environment obviates data implementation concepts and security considerations. Moreover, it is updatable; precision of architectural decisions and quality of designed architectures will improve by time.*

**Keywords:** architecture style, heterogeneous style, Decision Support System, fuzzy inference.

## 1 Introduction

Nowadays, fundamental role of software architecture as a powerful contrivance in controlling complexity of projects is such unavoidable that architectural design is an essential part of development lifecycle [1]. Making use of software architecture styles and patterns is a most common way in architectural design which helps in finding risks early and increasing quality of products [2]. Architecture styles and patterns are some reusable and frequently-used structures which specify software components, their specific properties and relations among them [1, 3]. However, in spite of having some recognized advantages and disadvantages, these results are not conclusive and should be refined in different domains or in case of having combined styles [4]. Today, due to the enormous increase in terms of complexity and scale of projects, the importance of this matter increases [5].

Consequently, problem of selecting architecture styles is a multi-criteria problem [6] in which lots of features of a project should be considered. Aggregation of all these criteria is a very complicated and challenging issue with

which most of human users try not to be faced. Hence, we need tools that are able to first, consider all criteria related to the problem domain to aggregate them, and then, select suitable styles and patterns which can perfectly cover different aspects of the domain as much as possible [7]. To overcome mentioned issues, an environment is represented in this paper which is a set of tools capable of: storing and retrieving all necessary data and information, making inferences, making multi criteria decisions to select and design heterogeneous architecture styles, and suggesting some alternative architectures for the system with respect to architect's priorities.

Moreover, because of updating capability, expertise of the environment will increase after being used in different projects. The abstract design of the DSS and the needed tools were presented in [6, 7]. In this paper, architecture of data communication and existing access levels are presented in the implementation of the DSS which prove capabilities of the DSS in practice.

The remainder of this paper is organized as follows. Issues we faced in selecting a suitable style are discussed briefly in section 2. Existing composite styles are classified in the third section. Moreover, these two early sections contain some considerations we faced with while implementing the environment. Some details about DSS of architectural selection are presented in section 4. Data structure and constitution of interaction between users and DESAS are topics of scrutiny in the fifth section. Section 6 includes related work. Last section contains conclusion.

## 2 Issues of Style Selection

Selecting suitable architecture style(s) that can help us in satisfying functional and especially non-functional requirements of a system correctly and precisely is one of the most important parts of software design process [8]. In order to select suitable architecture styles, that are able to cover different characteristics of the problem domain and satisfy wide variety of requirements, different goals and objectives should be considered. Hence, architecture style selection is a multi-criteria decision-making problem. In

addition, comparing capabilities and benefits of software architectures is somehow difficult; moreover, results should be refined and completed in accordance with architects' experiments [6]. Additionally, complexity of architecture style selection will increase as we need to aggregate effectiveness and importance of each requirement in order to make precise decisions.

When deciding about architecture styles to select, based on the problem domain, selection of only one style among existing styles, including simple or heterogeneous styles, might satisfy all requirements. In this case, only functional and non-functional system requirements and priorities of the architect are taken into consideration. But in some cases, only one of the existing simple or heterogeneous styles does not satisfy requirements and cannot cover the problem domain completely [9]; therefore, more than one of the existing styles must be selected. In this case, not only functional and non-functional requirements and priorities of the architect should be considered, but also *combination constraints* should be taken into account.

Making use of more than one architecture style comes with consistency problems and constraints. In this situation, by considering each architecture style alone, including heterogeneous or simple, different results might be obtained in comparison with the case that each one is part of (another) heterogeneous architecture and should be combined with other style(s) [9]. For example, when an existing style which satisfies a quality attribute, e.g. performance, significantly, is combined with, e.g. embedded into, another style which satisfies performance as well, obtained architecture will not necessarily satisfy performance significantly too. Level of satisfying performance by overall architecture must be evaluated.

Besides, because of the complexity of today's software systems and their large scale, most of systems should be developed based on heterogeneous architectures which are combination of different architecture styles. For selecting architecture styles, evaluation methods and techniques [10] are usually used; but these methods do not pay attention to the abilities and capabilities of styles.

### 3 Possible Composite Styles

In order to implement an environment which is able to design software architecture, we need to know possible combinations of architecture styles. Because, combining architecture styles in order to cover the problem domain is one of the most important issues in architecture design. Generally we can classify these combinations into four categories [11] (imagine  $p$  and  $q$  are architecture styles):

- Sequential heterogeneous styles
- Embedded heterogeneous styles
- Parallel heterogeneous styles
- Hybrid heterogeneous styles

*Sequential-* Putting architecture styles together in a way that when one part of system, which has a special style, finishes, another part with a different style will start.  $p \nabla q$  indicates sequential arrangement of architecture styles.

*Embedded-* Whenever a component of an architecture style has another style itself, it is called embedded heterogeneous architecture style.  $p \Delta q$  means that  $q$  is inside  $p$ .

*Parallel-* Whenever two or more architecture styles exist in a system structure without any interaction, we have parallel heterogeneous styles. We represent it by  $p || q$ .

*Hybrid-* By virtue of complexities of contemporary systems and their need for making use of styles that have enough coverage of the problem domain, sometimes we should use a combination of the mentioned styles.

## 4 DSS for Architecture Selection

The DSS defined in [7], is a meta-model that can help architects to choose suitable architecture style(s) and design architecture of their systems. This DSS takes advantage of all useful information that could help in performing precise decisions [7]. It includes of four essential components which give the system abilities of storing, extracting, and adding all necessary information. These components are: Knowledge base, Tools, Decision maker, and User Interface.

*Knowledge base:* knowledge base is the most essential component of the system and contains all necessary information needed for making precise decisions. It contains three components: domain repository, style repository, and rule base. *Domain repository* contains information related to the importance of each quality attribute in different domains. This information could be updated after each decision making process; update is performed with respect to the new results in various domains. *Style repository* contains information related to the level of satisfaction of different quality attributes by different styles. Generally, style repository includes mechanisms of storage, search, and update. It maintains styles and patterns and, moreover, their categories, the relation among them and experimental information about their usage. *Rule base* contains some rules which indicate interaction among quality attributes in an architecture style with respect to the domain. These rules are extracted from architecture styles repository and domain repository.

*Tools:* this component is a collection of some tools which are needed in the DSS. Aggregation tool is the most important tool which is used to aggregate different criteria; and its precision has a direct relation with quality of results. Hence, a fuzzy aggregation tool is used in the DSS which has an acceptable precision and has proved its capability in architecture style selection [6]; however, other tools could be used for aggregation. Extracted rules and priorities of architect are inputs of the fuzzy tool.

Moreover, other tools can be used in line with aggregation tool which will be discussed in the next section.

*Decision maker:* decision maker is another important component of the system. The responsibilities of the decision maker are: receiving and sending information from and to all components of the DSS, recognizing new compositions and adding them to the styles repository, and making decision about updates. Moreover, this part consists of an internal human agent as an internal expert architect and makes decisions about updates [7].

*User Interface:* generally, receiving users' input, included all necessary information to make decision, and delivering it to the decision maker; receiving decision results and representing it for users; and retrieving some information from system are performed by this component.

But, implementation of these components, the way of interaction with users, and the way of handling this interaction by exploiting expertise factor are discussed in the next section.

## 5 DESAS V1.0

Describing each component of the DSS, in order to exploit their benefits and capabilities, is important and, if performs suitably and appropriately, will increase precision and validity of decision results. Hence, in order to implement the DSS efficiently and effectively, some requirements should be satisfied. For this system, three main requirements are considered: issues in knowledge base implementation, validation of entered data, and customization for each user. Satisfaction of requirements and sub-requirements can provide us with benefits and capabilities of the DSS in selecting and designing architecture styles, especially composite styles. Moreover, other facilities are considered for the environment which increase its capabilities and improve its applicability by specific updating process. These facilities are mentioned in a separate subsection.

### 5.1 Implementation of Initial Knowledge Base

To design the knowledge base in a way that increases applicability and quality of designed DSS some important requirements are concerned. At first, structure of repositories and rules must be inferable and must consider different aspects like quality and quantity attributes, and interaction among them so that high quality inferences are performed. In works presented in [6, 7], theoretical structure of databases and tables of the DSS was represented. But we noticed that in order to implement the knowledge base perfectly, other considerations like notations to formalize different combinations of architecture styles [11] must be concerned. Moreover, we need rules which can determine interaction among quality attributes of an architecture style with respect to the specified domain and specified importance level of all possible combinations of criteria.

As mentioned, rules are stored in the rule base and are extracted from style and domain repositories. These rules cover all aspects of interaction among criteria sets, including synergy and redundancy, with respect to the problem at hand. In the designed rule base two types of rules exist: one for interaction among styles, while the other covers interaction among quality attributes. Equations (1) and (2) are hold for each type respectively.

$$\text{IF } L \Delta P \text{ THEN } (Q_s)_i = Q_s \quad (1)$$

$$\text{IF } Q_c > T_1 \text{ And } Q_p > T_2 \text{ Then } Q_p = Q_p + e \text{ AND } Q_c = Q_c + e \quad (2)$$

Translation of (1) with respect to the notation defined in [11] and mentioned in section 3 is: if the pipes-and-filters architecture style (represented by P) embeds into layered architecture style (represented by L), security attribute of the overall system will not change. General translation of (2) is: if a specified architecture style, concurrently satisfies two quality attributes cost (represented by  $Q_c$ ) and performance (represented by  $Q_p$ ) more than the threshold  $T_i$ , extra profit is obtained.

Additionally, rule selection can be categorized in two types of simple and complicated. If requirements domain that should be satisfied is not extensive, selection of a rule would be enough. In this case, the rule exactly exists in rule base and is enough for the condition. Therefore, making use of fuzzy integral, which is a discrete tool, is enough for decision making [6]. But in some cases, selection of an existing rule cannot support the condition and more than one rule is needed. In this case, selected rules and priorities are used to make fuzzy inference.

### 5.2 Data Validation

Data validation is an important requirement of system implementation that should be covered. If a DSS user, changes some information related to the repositories correctly or not, this change must not affect the main repositories of the system before being certified. To obviate the need, all performed changes enter to a separate place and will not affect main data. It is the responsibility of the expert architect to take these changes under close scrutiny and perform them into the main database in case of being certifiable. Another important thing is that the expert architect can modify these changes before storing to the database. Moreover, he can modify data stored in the database if needed. These actions are performed through a user interface component called EXA form and represented in Figure 1. Modifications can be performed with respect to expert's experiences and results of previous projects which have been performed by making use of the DSS. Besides, expert architect is responsible for validating new composite styles and adding them to the main database. Although the responsibility of initializing the knowledge base is up to the expert architect, but dependability to the expert will diminish by increasing the expertise level of the system; this is a case of having change in the architecture of DSS after long usage.

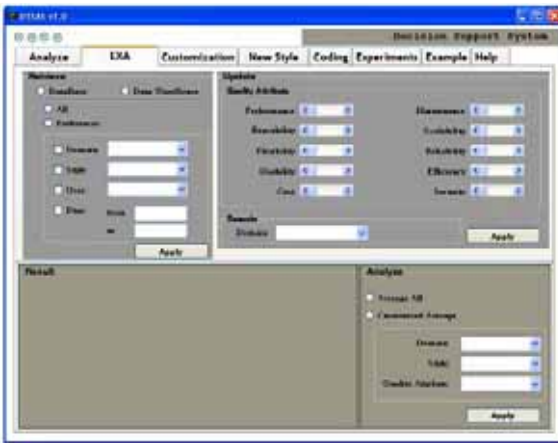


Figure. 1. A schema of the EXA form

### 5.3 Customization for Each User

Although all changes that each user performs do not enter to the repositories directly, but every user must be able to retrieve these changes while accessing to the main data. Hence, the place to store empirical data should be nonvolatile and the users must be able to retrieve previously-entered data and change this information if needed. Consequently, to obviate these requirements, we make use of layered architecture for interaction of user with the environment which is represented by Figure 2. This structure is designed for the access of users to the data, and moreover, represents internal structure of the user interface component. This multi-layered architecture avoids occurrence of repugnance and avoids entrance of invalid information. Moreover, it provides the DSS with more flexibility and the ability of being customized for each user. Therefore, users not only can exploit the main data of the system but also can make use of their stored experiences and opinions in decision making processes.

As represented in Figure 2, a central database exists in the lowest layer which consists of style and domain repositories' data. The decision maker component of the DSS has a human agent -expert architect of the system- which has the authority of changing these data and updating repositories. In order to differentiate users, and satisfy security requirement, four access levels are defined based on users' experiences. Abilities of updating, inserting, and adding new styles are motivation for differentiations. For example, the internal expert architect has complete access to all parts of the system. By defining access levels based on experiences we can rank users' results and opinions. The access control information resides in the central database.

In the next level, a data warehouse is considered which consists of lower data by considering users' customized and changed data; in addition, temporary and empirical data of users will not affect the main data. It is used to store static analyses and perform dynamic analyses; the request of a dynamic analysis is received via user

interface. Additionally, users can perform some changes like updating information of each style, adding a composite style, or adding a new style into data warehouse, with respect to their access level.

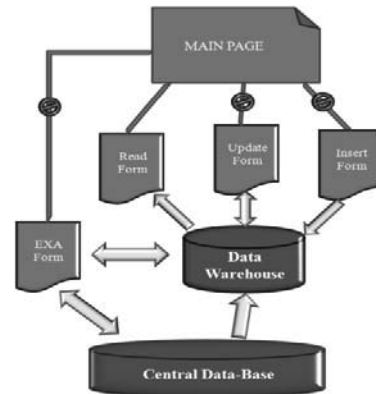


Figure. 2. Structure of user interaction

Analyzing and searching for architecture styles should be performed with respect to both needed quality attributes and project domain. This capability is provided for users via analyze page of the user interface component which is represented in Figure 3. Moreover, user can choose the place to perform search (either customized individual data in the data warehouse or the base-data in the main database) and compare results to take the best decision.

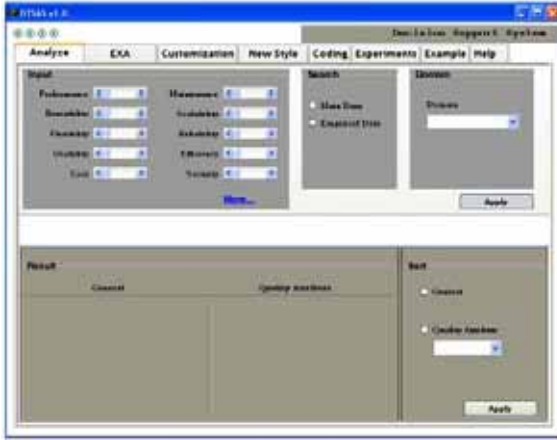
### 5.4 Other Facilities

The DESAS v1.0 is a first release of our tool and to complete it and to help in having better performance and more usability we should perform progress in accessories and more parts of the environment. For example a coding part is included which can help in implementations; because codes are predefined and we can take advantage of this capability. In the experience part some experiences about styles and their composition deployed on different domains are included. Other parts and useful accessory features will be completed in future work.

### 5.5 Case Study

In order to represent an experience with usage of the tool, we take advantage of the famous KWIC example [8]. To compare different architecture styles we take advantage of three criteria: performance, reusability, and flexibility. We incorporate criteria *change in algorithm* (CA), *change in data representation* (CD), and *change in function* (CF) into the criterion *flexibility* to increase precision.

As represented in Figure 3, in the input part, there is a "more" link by which user can use more criteria related to the problem domain; moreover, he can just use the fuzzy inference tool from this part. Note that what is shown in the GUI are some common quality attributes. It is the responsibility of fuzzy aggregation tool to calculate importance of flexibility. In order to obtain result, two rules (3) and (4) are extracted from rule base.



**Figure 3.** A schema of the analyze page

$$\text{if } Q_{CF} > T_1 \text{ And } Q_{CR} > T_2 \text{ Then } Q_{CF} = Q_{CF} - e \text{ And } Q_{CR} = Q_{CR} - e \quad (3)$$

$$\text{if } Q_{CA} > T_1 \text{ And } Q_{CF} > T_2 \text{ Then } Q_{CA} = Q_{CA} + e \text{ And } Q_{CF} = Q_{CF} + e \quad (4)$$

Once obtaining a result for flexibility, overall architecture of the system must be selected by considering all quality attributes. In this case, some other rules are extracted from rule base which represent interaction among quality attributes. When this phase is performed, an evaluation of each architecture style is obtained which represents level of adequacy of each one for the problem at hand. Obtained results are used as basic information for entering to the database again. Next, system architect should consider the limitation of *implementation cost*, which generally is an effective and important parameter, in order to balance technical requirements and costs. Hence, the system will extract information about cost of architecture styles and starts analyzing. Rules (5), (6) and (7) represent some extracted rules during inference.

$$\text{if } L \Delta \text{ ADT Then } (Q_P)_i = \text{Min}((Q_P)_L, (Q_P)_A) - e \quad (5)$$

$$\text{if } L || I \text{ Then } (Q_c)_i = Q_c + (OD_i)_L \times |(Q_c)_L - (Q_c)_I| \quad (6)$$

$$\text{if } P || B \text{ Then } (Q_c)_i = Q_c + (OD_i)_B \times |(Q_c)_P - (Q_c)_B| \quad (7)$$

The notation  $OD_i$  indicates the importance level of different parts of the system; With respect to the structure defined in [11].

Rule (5) says that making use of ADT embedded into Layered diminishes overall performance ( $Q_p$ ). Rule (6) says that making use of Implicit Invocation in parallel with Layered is useful to diminish costs and will reduce the overall cost. Rule (7), is same with (6). It is extracted because we imagine that, with respect to the fuzzy inference, Layered style was removed from candidate styles since it was not technically acceptable. Although, with respect to the fuzzy inference [6], Pipes-and-Filters is a valuable style for the problem at hand, but making use of Blackboard in parallel can balance overall cost; and it can be considered as a suitable design for this problem.

## 6 Conclusion and Future Work

In this paper, with respect to the approach that proposed to make use of a DSS, we designed an environment by which

some tools are implemented for selecting and designing heterogeneous architecture styles. This environment considers all criteria related to the problem domain and its expertise increases after each usage. It is secure enough and prevents entrance of invalid data; moreover, it can be customized for each user. In the designed environment information entered by users is confirmed and registered with respect to their experiences in projects. Furthermore the tool is client capable, so that user specific data can be processed in addition to the common and shared data. By exploiting unremittingly-updating information, expertise and performance of DSS will increase. The designed multi-layer architecture makes the environment capable of storing historical data by considering the time dimension. This data is used by expert architect to decide about changing information of the central database. Moreover, each user can store and retrieve his empirical data and exploit it in future without affecting the base knowledge of the system. Finally, we mention that this architecture for the DSS can be modified after several usages.

Representing an extended knowledge base to achieve more adaptation with software architects' working requirements is our future objective in order to complete the environment. This will obtain by exploiting the results which have acquired by making use of the tool practically.

## 7 References

1. Bass, L., Clements, P., Kazman, R., Software Architecture in Practice, Addison-Wesley Professional, 2<sup>nd</sup> edition, 2003.
2. Klein, M., Kazman, R., Attribute-Based Architectural Styles, Tech. Rep. CMU/SEI-99-TR-022., Carnegie Mellon University, School of Computer Science, 1999.
3. Tao, L., Fu, X., Qian, K., Software Architecture Design: Methodology and Style, Stipes Publishing L.L.C., 2006.
4. Svahnberg, M., Supporting Software Architecture Evolution- Architecture Selection and Variability, Ph.D. Thesis, Blekinge Institute of Technology, DS No. 2003:03, 2003.
5. Shaw, M., Clements, P., The Golden Age of Software Architecture, IEEE Software, Vol. 23, No. 2, pp. 31-19, 2006.
6. Moaven, S., Habibi, J., Ahmadi, H., Kamandi, A., A Fuzzy Model for Solving Architecture Styles Selection Multi-Criteria Problem, In proc. of 2<sup>nd</sup> UKSim European Modelling Symposium on Computer Modelling and Simulation, England, pp. 388-394, 2008.
7. Moaven, S., Habibi, J., Ahmadi, H., Kamandi, A., Decision Support System for Architecture-Style Selection, In proc. of 6th Intl. Conference on Software Engineering Research, Management and Applications, pp. 213-220, 2008.
8. Shaw, M., Garlan, D., Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall, 1996.
9. Firesmith, D.G., Capell, P., Hammons, C.B., Latimer, D., and Merendino, T., The Method Framework for Engineering System Architectures, AUERBACH, 2008.
10. AT&T. Best Current Practices: Software Architecture Validation, Internal report, 1993.
11. Moaven, S., Kamandi, A., Habibi, J., Ahmadi, H., (in press). Towards a framework for evaluating heterogeneous architecture styles, Asian Conference on Intelligent Information and Database Systems, 1-3 April, 2009.



# Towards Architecture-centric Collaborative Software Development

Yanchun Sun, Hui Song, Wenpin Jiao

*Institute of Software, School of Electronics Engineering & Computer Science, Peking University,  
Key laboratory of High Confidence Software Technologies, Ministry of Education,  
Beijing 100871, P.R.China*

*E-mail: {sunyc, songhui06, jwp}@sei.pku.edu.cn*

## Abstract

*In the development of large and complex software systems, software engineers are required to cooperate with their efforts. They develop shared understanding surrounding multiple artifacts, each artifact embodying its own model, over the entire development process. How to support software artifacts based collaboration efficiently within a large development process becomes a big challenge in software engineering. Since software architectures are considered as the blueprints for target software products and they can be used to organize various software artifacts in software development process from a high level perspective, this paper puts forward an architecture-centric collaborative software development approach to supporting the collaborative software development across the whole software lifecycle. The paper also illustrates how the approach works by studying one case in detail.*

## 1. Introduction

To produce large software, software engineers often develop multiple shared artifacts over the entire development process [1]. How to support software artifacts based collaboration efficiently within a large development process becomes a big challenge. This is just the main concern of this paper.

Software artifacts based collaboration distinguishes collaboration in software engineering from such broader collaboration that tends to provide artifact-neutral coordination technologies and toolkits. Artifact-neutral coordination technologies are based on natural language and easy to use but they are short of semantic information support for the development artifacts and context, which is easy to lead to ambiguous understanding for developers. On the contrary, artifacts based collaboration can solve this problem if we can use the models besides the artifacts very well according to the models' good structure, clear syntax and explicit semantics. Moreover, artifacts based collaboration can reduce the intensive

collaborative communication based on the common understanding of software artifacts.

For the collaboration based on software artifacts, the version control systems are often frequently used to manage the software artifacts [9]. These tools are very important for collaborative development among software engineers, but the artifacts stored in these tools are almost code-level programs and lack a reasonable organization from the viewpoint of software development process. As a result, these tools are short of support for collaborative development process.

To support the collaborative development, it is necessary to provide software developers with an appropriate model to organize various software artifacts in software development process from a high level perspective. Then, software engineers can collaboratively develop software based on this model.

With software becoming large and complex, software architecture (SA) becomes a blueprint to guide the development and maintenance of software systems. Some SA-centered development methods have been put forward for collaborative development [2,3,4,5,6,10], but they do not use the semantics of SA adequately, they just support simple collaboration for several authors based on the management of authority, and moreover, they support collaborative development just in special phase rather than the whole lifecycle.

In this paper, we put forward an architecture-centric collaborative development approach, which extends our previous approach [12] for supporting collaboration from the design phase to the whole lifecycle. First, based on version control tool and semantic information of SA, we abstract the information of fine-grained modifications into SA in order to support the collaborative design of software architecture among designers. Because SA is a core artifact in the whole software lifecycle, by introducing bi-transformation technologies [8], we transform the modification manipulation of other artifacts to the modification manipulation of SA model, to support the collaborative development among different developers.

The rest of this paper is organized as follows. Section 2 presents some related work. In Section 3, we

put forward an architecture-centric approach to supporting collaborative software development. Section 4 illustrates the approach by studying one case in detail. Section 5 concludes the contribution of the paper and gives the future work.

## 2. Related work

Software engineers in the academy and industry have developed a wide range of SA-based technologies to support collaborative work on their projects.

In the academy, Richard Taylor and David Garland present their own Architecture Description language (ADL) and propose the SA-centered development method based on the ADL [2,3]; ArchStudio from UCI [5] and ACMESstudio from CMU [7] typically support collaborative authoring by versioning architecture description files. MolhadoArch system from University of Wisconsin is integrated with a fine-grained version control tool to afford the collaboration at the level of individual model elements [6]. In the industry, Siemens's Hofmeister etc. describe a set of architecture views and put forward a corresponding software development method from requirement to implementation [4]. IBM also focuses on SA-centered development method and "Rational Software Architect" is an UML modeling tool focused on software architecture [10]. Engineers work collaboratively on diagrams with collaboration mediated via the configuration management system.

Compared with our architecture-centric collaborative development approach supporting for the whole software lifecycle, most of the collaborative supports provided by these tools above are fine-grained and without the semantic information of SA and moreover, they are just limited in the special phase rather than the entire software lifecycle.

## 3. An Architecture-centric Collaborative Development Approach

We present an approach supporting architecture-centric collaborative development, which is described as Fig. 1. Our approach is based on a software reuse methodology ABC(Architecture Based Component Composition) [11]. ABC method argues that SA should play a centric role in the whole software lifecycle. Based on this, we present the idea of the architecture-centric collaborative software development. During the whole development process, software developers manipulate and produce different artifacts in different phases. These artifacts have the relationships of refinement, and can be viewed as different views of software architecture model in some sense. Thus, our approach can support artifact based collaboration via introducing bi-transformation

between different artifacts and SA model.

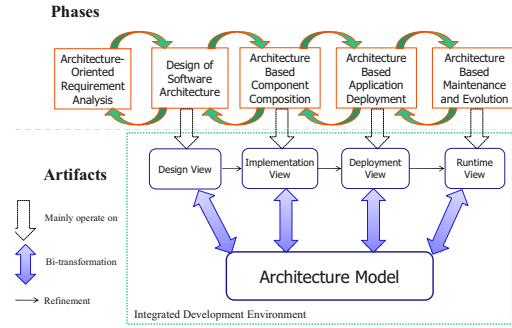


Fig. 1. Overview of the Approach

By referencing the typical architecture description language (ADL), we define the meta-model of software architecture by using Eclipse Ecore. Fig. 2 describes the core elements in the meta-model. The core concept of this meta-model is Component. We partition and organize every software system into components, each with a relatively individual concern.

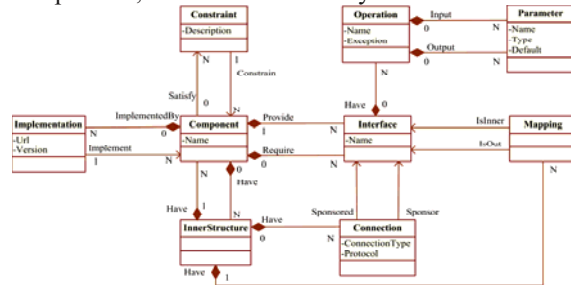


Fig. 2. Software Architecture Model in Our Approach

We also introduce the concept of *InnerStructure*, which helps organizing the whole system as a hierarchical structure to support the stepwise refinement during architecture design. Based on the meta-model, we construct a software architecture modeling environment by using Eclipse GMF, named ABCTool, in order that we could assist designers to record their design decisions by recording their manipulations such as additions and deletions of elements and modifications of properties and relationships of elements. The entire architecture model is recorded in the form of XMI in several files.

Designers can modify the SA model via the graphic interfaces. When the designers finish and save the modification, some related XMI files will be changed. As pure context files, these XMI files can be managed by a version control tool. In our work, we select CVS to achieve this. By using CVS, we can record who makes the modifications and what modifications have been made to SA model in the collaborative development process. We use Eclipse Plug-in to encapsulate the record file of these modifications and visualize them in ABCTool.

By CVS interface, we can obtain the information about the modifications from the XMI files. By analyzing the modifications information, we can elicit which elements in SA model have been modified and what kind of modifications have been made. Moreover, we can display the modifications explicitly in ABCTool, for example, using different color to distinguish added components, deleted components and unchanged components.

For the changed model, collaborative developers can select to accept, reject or add new modifications. The maintenance activities for modifications can be mapped to the operations in CVS. During the maintenance, collaborative developers can use the modification information offered by CVS to identify the intention of modifications. Sometimes, they may need to contact directly the developers making modifications to discuss the goals of the modifications.

In different phases, developers will deliver different artifacts, but most of these artifacts record some core information of SA. In other words, some transformation relationships exist between these artifacts and SA model. Thus, by transforming the core information in SA and adding special information in a given phase, the artifacts in the given phase can be constructed. Using those research fruits in the bi-transformation field [8], we can use a set of transformation rules to reflect the modifications of SA model into other models, and also reflect the modifications of SA level information in other models to SA model. Thus, we can utilize the approach above to assist with the collaborations among a variety of developers participating in different phases.

#### 4 Case Study

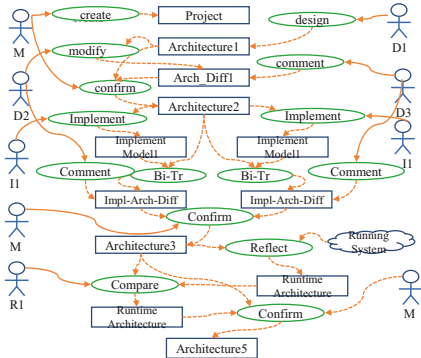


Fig.3. The Collaboration among Different Participants

In this section, we use a case to demonstrate our approach to collaborative development process. The case is about designing and developing a prototype website to support comparison shopping, based on two existing shopping system, Java Pet Store (JPS) and Rubis auction system. The comparison shopping system retrieves information of the same good from

JPS and Rubis and lists it on one page so that customers could make decision on where to shop.

There are 6 members participated in this case study, including a manager(M), 3 designers( D1,D2,D3), an implementer(I1) and a runtime administrator(R1). The detailed collaboration among these participants is described as Fig.3.

##### (1) Design Phase

First of all, manager M records the information about all members and launches this development process by creating a new project. Then he submits this project so that it could be used as a shared work space for all members. D1 is in charge of the big picture of this system. After an investigation of the existing systems, he decides to choose JPS and Rubis as a base to develop this comparison shopping system, and he records (Design) this decision into the first version of the software architecture. The current version of architecture only represents that the system must contain a UI component for comparing information about commodities, and the information is acquired from two existing systems, i.e. JPS and Rubis. Currently, the two systems are treated as single components(described as Fig.4).

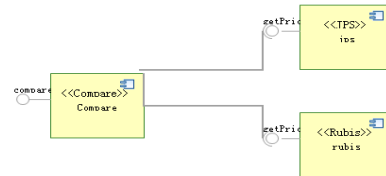


Fig.4. The First Design Version Made by D1

After the manager commits this version of architecture, D2 enters the design activity by checking out this shared project along with the first version of architecture model. D2 notices that it is not a good design for subsequence maintenance that the information retrieving logic, the comparison logic and the user interface in the original version of architecture are all encapsulated in component “Compare”. Thus he separates the original component “Compare” into three ones and get a new version of architecture. As a common designer, D2 does not have the authority to commit his architecture as a new version, but he can compare his version with the current version in the project, which is the first version designed by D1. The comparison version is shown as Fig.5. The green background means new added component and the blue background represents the modified ones.

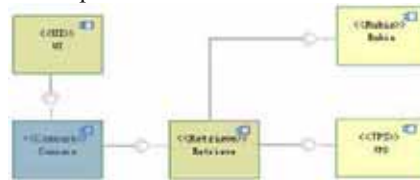
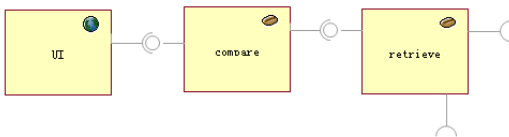


Fig.5. The Comparison Version Made by D2

All the members of this development team can check out this comparison version to find what has been changed by D2. They can also make their comments on the changes. Now D3 enters the project, and from the comparison version she knows that D2 added a new component “UI” and a new component “Retrieve” and changed the component “Compare”. She agrees with D2’s decision of providing separated component for retrieving information about commodities, and thus she records “agree” as well as her comments on component “Retrieve” in comparison version. But in the meantime, she thinks that the logic of “comparison” is not complex and does not need a separate component. So she records “disagree” on component “UI” and gives her comments. She also provides her own version of architecture, and commits the comparison version. Now every member of this team can check the two comparison versions, and give their comments on the changes. Finally, M checks the two comparison versions, collects the comments, and makes the final decision. In our case, M finally adopts D2’s version, and merges (confirms) D2’s change with his original version. Then he commits this new architecture model as the second version of architecture design.

**(2) Implementation Phase**

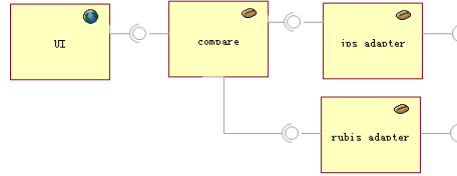
The team uses this second version of architecture model to start implementation. I1 first uses our transformation support to systematically translate the architecture model into an implementation model as follows. As JPS and Rubis are all J2EE applications, I1 also chooses J2EE as the platform for his implementation. Thus the implementation model(described as Fig.6) is specific to J2EE. I1 implements “UI” as a web component, and implements component “compare” and “retrieve” as two individual EJB components. EJB “retrieve” interacts with the existing JPS and Rubis system via remote procedure invocation (RMI).



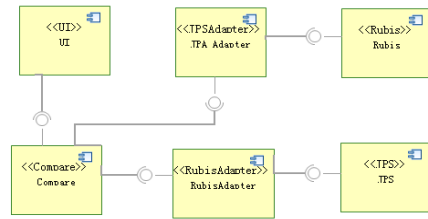
**Fig.6. Implementation Model Made by I1**

This implementation model directly complies with the second version of architecture model. But when considering deployment, I1 finds that this implementation is not satisfied. He notices that there are a big amount of data exchange between component “retrieve” and the existing JPS and Rubis system, and thus RMI will cause bi-performance penalty. So it is better to implement two EJBs for retrieving and adapting data, so that he can deploy the EJB for JPS

onto the same server with JPS system, and deploy the retrieve EJB for Rubis on Rubis’s server.



**Fig.7. Implementation Model Made by I1 when Deploying**



**Fig.8. The Third Version made by M**

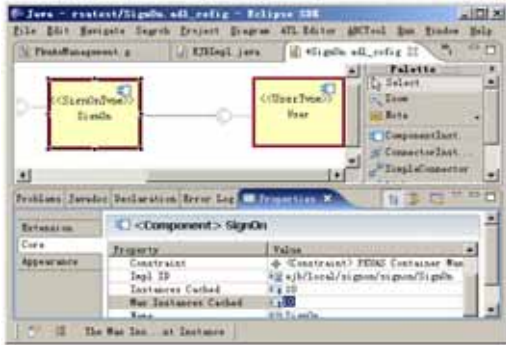
But this implementation does not comply with the final architecture, and as an implementer, I1 does not have the authority to arbitrarily change the original design. Now he can use bidirectional transformation to trace back his change, and get a new architecture model. Then he compares his trace-back architecture with the original architecture, and commits the comparison version. Finally, M checks out the comparison version of I1’s architecture, and notices that the only difference is that the retriever component is substituted by two adapters. He thinks that this difference does not conflict with the original decision of the designers, and thus M adopts I1’s modification and commits it as the third version of architecture(shown as Fig.8).

**(3) Runtime Maintenance Phase**

In the current version of architecture, JPS and Rubis are all composite components. In maintenance phase, the runtime administrator needs to use the detailed information, including the inner structure, the information about running platform, and the runtime data collected from the platform, to maintain the system at runtime. Our comparison shopping system is deployed on a J2EE compliant application server, named PKUAS. In this section, we present how R1 uses our architecture-based support to maintain the running system.

The Fig.9 is a snap shot of our architecture-based runtime management tool for PKUAS. The central editor shows part of the architecture model of JPS. Many of the elements in this architecture model are actually the images of resources at runtime. For example, the component SignOn is an image of the EJB named SignOnEJB running on PKUAS, and some of the attribute values displayed in the bottom attribute view are actually retrieved at runtime. R1 can make his diagnosis for the running application on the basis of

the attribute changes or element emergences or disappearances, and he can manipulate the running application by directly changing some of the attribute values, e.g. he can change the value of MaxInstancesCached, and the instance pool allocated for SignOnEJB will be resized, when the maintainers finally launch the synchronize command.



**Fig.9 Architecture-based Runtime Management Tool**

After the system has been used for some time, R1 notices that the early version of JPS does not block brute force attacks, which means a malicious user may access the store by trying passwords for many times. R1 knows that it is caused by inadequate constraints on the component SignOn, so he instantiates a new Constraint element in architecture model, assigns it with a simple prototype implementation, and inserts it into the constraint list of component "SignOn". When he launches the synchronization command, the interceptor will be dynamically inserted into the container of "SignOnEJB" without restarting the server. Since all the changes on the system should be decided by M, R1 also has to trace back his changes into original architecture model. As a local modification on an inner EJB inside the JPS system, this modification only affects the inner implementation of JPS component in the original architecture model, without changing its interface. Therefore, M concludes that this modification does not violate the original architectural decision, and he adopts R1's runtime evolution.

## 5 Conclusions

This paper puts forward an approach to supporting architecture-centric collaborative development in different phases. We use SA model to distill the semantics of context changes recorded in CVS, and then display the change of SA model to assist different designers to collaborate their design. By using bi-transformation technologies, we use the transformation relationships between SA and other artifacts to support the collaborative development for different developers.

In the future, we will make further research on how to introduce more architectural knowledge (e.g., design rationale) to facilitate the collaborative development.

**Acknowledgments.** This effort is sponsored by the National Basic Research Program of China (973) under Grant No. 2009CB320703, and the National High-Tech Research and Development Program (863) of China under Grant No. 2007AA01Z127, 2008AA01Z139, and the Science Fund for Creative Research Groups of China under Grant No. 60821003.

## References

1. Jim Whitehead, "Collaboration in Software Engineering: A Roadmap", In: Future of Software Engineering(FOSE'07), Minneapolis, MN from May 19 to May 27, 2007.
2. Nenad Medvidovic, David S. Rosenblum, Richard N. Taylor, "A language and environment for architecture-based software development and evolution", in Proceedings of the 21st international conference on Software engineering, Los Angeles, California, United States, 1999.
3. David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, Peter Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure" , Computer, vol. 37, no. 10, pp. 46-54, Oct., 2004.
4. C Hofmeister, R Nord, D Soni, Applied Software Architecture, Addison Wesley, 2000.
5. UCI Software Architecture Development Environment, 2007, <http://www.isr.uci.edu/projects/archstudio>.
6. T.N.Nguyen and E.V.Munson, "Object-oriented Configuration Management Technology can Improve Software Architectural Traceability", in 3<sup>rd</sup> ACIS International Conference on Software Engineering Research, Management and Applications(SERA'05), Mount Pleasant, Michigan, USA, 2005, pp.86-93.
7. A. Kompanek, "Modeling a System with Acme", 1998, <http://www.cs.cmu.edu/~acme/html/WORKING-%20Modeling%20a%20System%20with%20Acme.html>.
8. Yingfei Xiong, Dongxi Liu, Zhenjiang Hu, Haiyan Zhao, Masato Takeichi, Hong Mei, "Towards Automatic Model Synchronization from Model Transformations", in Proceedings of 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), Atlanta, Georgia, November 5-9, 2007.
9. Grady Booth, IBM Rational, "Introducing Collaborative Development Environments", Dec 2006, <http://www.alphaworks.ibm.com/contentnr/cdepaper>.
10. IBM, "Rational Software Architect Overview," 2007, <http://www-306.ibm.com/software/awdtools/architect/swarchitect/>.
11. Hong Mei. ABC: Supporting Software Architectures in the Whole Lifecycle. In: Proceedings of the Second International Conference on Software Engineering and Formal Methods (SEFM'04), 28-30 September 2004, Beijing, China. IEEE Computer Society 2004.
12. Yanchun Sun, Hui Song, Xinghua Wang, Wenpin Jiao. Towards Collaborative Development Based on Software Architecture. In the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'2008), Redwood City, CA, USA. July 1 - July 3, 2008, 250-254.

# ANALYSIS OF AGENT ORIENTED SOFTWARE ENGINEERING METHODOLOGIES FOR SOCIAL CAUSAL MODELS

Michele Atkinson, Kutta Technologies, Inc.  
Sheryl Duggins, Southern Polytechnic State University

*Abstract* -- This research examines the application of agent-oriented software engineering methodologies to a social causal model. It evaluates several popular agent-oriented software engineering methodologies including Gaia, TROPOS, Prometheus, and AUML. The research presents an analysis of the methodologies and focuses specifically on causal models, where agents are used to analyze a cause/effect relation between a social attitude and a social effect. A case study is performed and analysis is summarized as recommendations for improving agent-oriented software engineering methodologies.

## 1. INTRODUCTION

Social causal models are used to study the cause and effect relationships at play in social systems. Social scientists use causal models as abbreviated depictions of a realistic setting based on abstraction and idealization. Distinct from a replica (which aims to produce an identical copy of the social system), a model abstracts away details that are not of interest and assembles abstractions of what is interesting [1].

Agents are particularly well-suited for building social causal models because their inherent properties tend to emerge behaviors, as human behaviors do, in a macro-level effect of the micro-level agent interactions. Wooldridge and Jennings [4] described the properties of agenthood as: autonomy, reactivity, proactivity, and social ability. When agents possess each of these properties, they possess an individual reasoning ability that (when observed as a society of agents) evolves social outcomes that might not have been considered before. These evolutionary characteristics are the main reason analysts are interested in social applications of agent modeling. We wondered how much more accurate social models could be if engineering methodologies were improved so that analyzed causal relationships could be aggregated to project possible outcomes. With this research we hope to identify where subjective decisions exist in current methodologies and to suggest improvements for minimizing subjective decisions in the design process.

### *A. Rationale*

The U.S. Army is investing millions to improve their understanding of the causal system at work in the Iraqi insurgency. Insights need to be organized and disseminated as a social model of human intelligence data so that cultural lessons learned can be preserved across brigades, from one deployment to another. To harness the power of networked

intelligence data, the model will be distributed on the Joint Forces Global Information Grid as a powerful system of systems. Machine-readable human intelligence will be mined, analyzed, and distributed across the Grid with Semantic Web technologies including metadata, ontologies, and intelligent agents [2]. Agents are critical pieces of the Grid, because they are the system components that will fuse latent, incomplete, and inconsistent observations about people, events, and relationships into cohesive information. They will interpret and apply cultural rules to derive possible courses of action and mitigate undesirable effects. Finally, agents will learn and refine the uncertainty of the human intelligence as they analyze more data. In short, agents will help U.S. troops manage and propagate cultural insights from one brigade to another long after the first and second deployments have returned home. Because of this potential, agents are of great interest for modeling cultural understanding and tracking insurgents.

### *B. The Need for Agent-Oriented Software Engineering Methodologies Applied to Causal Models*

Causal models are social simulations that are used to evaluate whether there is a causal relationship between a given stimuli and an observed effect. The following points illustrate the difficulties that social simulations present.

**Belief systems are not static.** In the Iraqi insurgency, beliefs, desires, and intentions evolve as events unfold; therefore every possible permutation cannot be anticipated at design time. Methodologies need to account for the evolution of belief systems as events unfold.

**Group belief systems are aggregates of the individual belief systems.** The Iraqi insurgency is a perfect example of how the groups often have their own belief systems that emerge from separate and distinct individual belief systems. It is the group's integrated belief system that shapes the actions taken on the environment. Yet the individual belief system must still be represented explicitly because groups often splinter and form new groups. This implies an aggregation (and to some degree, a computable summation) of individual attitudes.

**Groups influence individuals.** Though lone suicide bombers have become a more common occurrence in some parts of Iraq, the suicide bombers are not acting alone. Their acts are generally planned and encouraged by a larger, organized group. Again, this implies that the group has a collective belief system that shapes the interactions on the environment as opposed to individual belief systems. The

group's belief system actually constrains the individual's actions on the environment.

The goal of this research is to identify which of the existing AOSE methodologies provide the most robust simulation while handling these unique challenges.

### C. Approach

This research will evaluate four AOSE methodologies, propose evaluation criteria related to social causal modeling, and evaluate each of the selected methodologies using those criteria. The methodologies selected for this research include Gaia [5,6], TROPOS [7], Prometheus [8], and Agent UML [9,10]. One of these methodologies will be chosen to implement a case study on the Iraqi insurgency and the results are presented here.

This paper assumes familiarity with the AOSE methodologies and will not describe them in detail here. The four models are representative of a larger classification of agent design approaches in the AOSE methodology literature, including Belief Desire Intent (BDI) models, societal role models, and agent-based UML models [12]. Gaia and TROPOS incorporate the societal role models, while Prometheus does not. TROPOS and Prometheus incorporate the BDI models, while Gaia does not. AUML supports agent-based UML models and provides no support for BDI constructs. An analysis across these models should give insight into whether causal models are better suited to societal role modeling, BDI modeling, or AUML modeling.

Each of these methodologies will be evaluated according to the criteria established in prior research [11,12] as well as new criteria specific to causal modeling. Highlights of our Iraqi case study will be detailed, followed by our recommendations for future work.

## II. MODEL CLASSIFICATIONS

Originally proposed by [13], BDI models use mentalistic constructs to represent the complex and dynamic aspects of the system. Before the idea of beliefs was introduced, the agent's environment was usually captured in rigid data structures called concept frames. However, beliefs introduced the concept of managing uncertainty about the environment. Beliefs are distinct from concept frames because they include a measure of uncertainty about the environment that influences how the agents act.

Desires represent the objectives of the agent. Because desires are expressed as desirable environmental states, agents can exhibit proactiveness, reactivity, and autonomy in pursuing the desired goal state without explicit direction from another entity. Intent is an agent's expression of available alternatives in pursuit of its desires. Expression of intent is a critical piece to enabling social ability because negotiation and coordination with other agents requires a vocabulary for expressing what needs to be done and how agents will organize tasks amongst themselves. Ultimately,

autonomy, reactivity, proactiveness, and social ability came to be the defining properties of agenthood [5]. Therefore, it is a reasonable expectation that mentalistic constructs may be a necessary to decompose agent systems.

Societal models focus on identifying the role each organization plays in a society. The social level analysis at the system layer identifies the domain's relevant organizations. The analysis treats those organizations as agents and analyzes their interaction with each other through goals and actions. Thus, the societal role model situates the domain's mentalistic decomposition inside the domain's societal decomposition.

## III: METHODOLOGY EVALUATION CRITERIA

Our evaluation criteria are derived from the work of [12], but the evaluation is our own work. Lin's group organized their criteria into four categories similar to those of [11], including: 1) concepts and properties, 2) notations and modeling technique, 3) process, and 4) pragmatics. The concepts and properties criteria relate to the desirable properties of the agent design and include autonomy, mental reasoning, adaptation, and social properties. The notations and modeling technique criteria relate to the methodology itself. These include whether the notation supports the required expressiveness, layers of decomposition, modularity, code generation, refinement of protocols, and traceability. The pragmatics criteria describe how practical a methodology is to use. Pragmatics take into account the availability of toolsets, the required level of expertise, whether a methodology assumes a particular implementation, and how easy it is to deploy agents. The Causal Modeling criteria are our own criteria and focus specifically on designing social simulations.

### A. Analysis

Our analysis scored each of the criteria for each methodology on an ordinal scale using an overall summation to evaluate which methodology was strongest in each area and overall. The observations are detailed in Tables 1-4. The scoring criteria have been omitted because of page limitations on this paper. In our analysis, AUML is compared along with Gaia since Gaia recommends using AUML for its detailed design phase.

Prometheus scored strongest in the concepts and properties criteria, because it is the most deliberate about addressing the organic components of agents. Prometheus explicitly designs the sensors and actions, as well as internal reasoning mechanisms for adaptation and autonomy. Prometheus pays particular attention to the social behavior of agents, guiding the derivation of their communications, collaborations, and organization.

**TABLE 1. CONCEPTS AND PROPERTIES CRITERIA.**

Gaia/AUML	TROPOS	Prometheus
<b>Sensed Perceptions and Interactions:</b> Sensed perceptions organized as resources early; interactions evaluated early and throughout.	Glosses over sensed perception design; protocols specified as AUML interaction diagrams.	Interface descriptions take place early on. The percept template descriptor describes the perceived data; action template descriptor describes actions.
<b>Internal Reasoning:</b> Represents the internal reasoning as a set of liveness rules and safety properties, but goals are not explicitly specified.	TROPOS represents internal reasoning as goals that are decomposed as capabilities and protocols.	Represents internal reasoning as goals that are decomposed as capabilities, plans, and actions.
<b>Adaptation:</b> Belief system adapts through the addition of new rules.	Adaptation requires a redesign of goal decomposition.	Adaptation requires a redesign of goal decomposition.
<b>Concurrency:</b> AUML specifically notates concurrent sequences in detail.	AUML specifically notates concurrent sequences in detail.	Prometheus does not provide guidance for concurrent designs.
<b>Communications:</b> Protocols are explicitly designed and associated with roles and activities.	TROPOS recommends using AUML to diagram agent protocols. Agent protocols are derived from the goal capability derivation.	Uses protocol and message descriptors to describe interactions between agents, which decompose into actions. More detail than others.
<b>Collaboration:</b> Goals implied in liveness rules and safety properties. Rules can be achieved collaboratively by sharing a protocol and using the specified data format to exchange necessary information.	The protocol definitions provide a way for agents to share goals. These are delegated to AUML.	Prometheus has a specific data template for shared data objects that is derived after the protocols are in place.
<b>Agent Abstraction:</b> Gaia refers to organizational patterns to guide agent partitioning. It recommends organizational efficiency and simplicity, but does not explicitly guide them.	Agent partitions are derived from stakeholder analysis.	Data coupling diagrams are used to analyze the coupling and cohesion between agents. Prometheus is the most detailed in this particular area.

**TABLE 2. NOTATIONS AND MODELING CRITERIA EVALUATION.**

Gaia/AUML	TROPOS	Prometheus
<b>Expressiveness:</b> Gaia’s formal notation is easier to maintain earlier on while agent partitions are still forming. AUML is the most expressive as far as diagrammatic notations go.	TROPOS relies on diagrams to decompose the system, but lacks the template definitions for each diagram entity found in Gaia and Prometheus. Without templates, TROPOS feels somewhat loose in definition and scope.	Prometheus’ template diagrams guide the designer through a detailed description of the agent system, including its belief system, its environment, and its communications.
<b>Complexity:</b> Gaia is somewhat vague about how to partition the agents, but AUML is extremely strong at nesting diagrams to conceptualize a system from general to specific.	TROPOS moves from stakeholder analysis to goals, then capabilities, and finally protocols. It is less of a modular approach and more of a waterfall of decompositions.	Prometheus starts with goals and decomposes more specific goals from there. However, Prometheus seems to be missing the most abstract layer, which is the role layer.
<b>Modularity:</b> Captures protocol abstractions for reuse.	No focus on reuse.	No focus on reuse.
<b>Executable:</b> Since no toolset exists at this point, there is no code generation.	Tools do exist for TROPOS, but none appear to generate agent code.	There does not appear to be a toolset to support Prometheus.
<b>Refinement:</b> Reasoning elements (rules) are derived from analyzing the capabilities needed for each role. AUML is particularly nice for refining Gaia’s more abstract specifications into	TROPOS actually derives goals first and backs them into a list of stakeholders. It seems more natural to derive the stakeholders first; allowing the goals to fit inside the	Prometheus skips over role identification and starts with goal identification which is decomposed into the belief system of capabilities, plans, and actions.



detailed sequences and collaborations.	scope of the identified stakeholders.	
--	---------------------------------------	--

**TABLE 3. PRAGMATICS CRITERIA EVALUATION.**

Gaia/AUML	TROPOS	Prometheus
<b>Tools:</b> Evaluation of AUML toolset being investigated at <a href="http://www.auml.org">http://www.auml.org</a> , but no toolset established as of August 2008.	A suite of tools is available at <a href="http://www.troposproject.org">http://www.troposproject.org</a> .	AUML has been integrated into the Prometheus toolset.
<b>Required Expertise:</b> Gaia relies on organizational modeling and patterns.	No background information required.	No background required. Mission statement is to cater to those unfamiliar with agents.
<b>Modeling Suitability:</b> Slanted toward rules based architecture.	Tied to BDI architecture.	Tied to BDI architecture
<b>Domain Applicability:</b> Formal notations and diagrams support any domain's concepts.	Diagrams support any domain's concepts.	Templates and diagrams support any domain's concepts.
<b>Scalability:</b> The formal notation predicates might be very hard to manage for a very large domain.	Diagrams can be scalable if they are handled in modular chunks.	Diagrams can be scalable if they are handled in modular chunks.

**TABLE 4. CAUSAL MODELING CRITERIA.**

Gaia/AUML	TROPOS	Prometheus
<b>Social experiment building blocks:</b> no support	no support	no support
<b>Aggregation of attitudes:</b> There is no decomposition for the computable summation of attitudes.	It is unclear how the semantic expression of an individual belief system morphs into a representative group belief system.	
<b>Decomposing uncertainty:</b> There is no decomposition of managing uncertainty in the sensed data.	no support	no support

Gaia/AUML showed best on the notations and modeling technique criteria, mostly because Gaia's formal notation suits early expressions without transforming them into operational rules or tree structures. AUML scored particularly well on the aspects of complexity and refinement because of its nested protocols notation.

There was no clear winner in the pragmatics criteria, showing that none of the methodologies have reached significant maturity in their toolsets or bodies of knowledge to dominate the industry. Unfortunately, TROPOS and Prometheus seem to be married to the BDI implementation without providing any kind of guidance to evaluate whether the BDI architecture is the right architecture.

None of the methodologies handle social experiment constructs directly. The scope of the null hypothesis and control and experimental variables, summation of attitudes, and the systematic decomposition of uncertainty are all conspicuously absent.

Overall, a combination of Gaia and AUML scored the highest on our criteria evaluation. At the time, we thought Gaia refrained from tying itself to a particular

implementation and AUML provided rich specification for implementation details. The formal notation seemed like a good solution for capturing scope without mapping onto operational constructs, and it seemed the societal focus of a social conflict would map well into Gaia. Therefore, Gaia was chosen as our methodology for the Iraqi case study outlined below.

*B. Case Study*

Our case study was based on Hashim's book [3] on the Iraqi insurgency. In the book, Hashim ponders a causal relationship between Sunni displacement and insurgency growth. Sunnis were generally thrown out of prominence in Iraq after the fall of Saddam Hussein. The Sunnis made up 22% of the population and were generally either educated professionals or highly skilled military veterans. The U.S. had promised stipends to disbanded military veterans that never came through. This led to a significant portion of educated people that were left idle and that shared a common disdain for the U.S. – a cast of willing hands ripe to be put to effective use in the insurgency.

A useful simulation would be to evaluate whether paying stipends to the Sunnis would have slowed the growth and maturity of the insurgency at this time. The following hypothesis is derived to guide the model:

**Causal model hypothesis #1:** *Paying the promised stipends to the Sunnis would have decreased their numbers in the insurgent ranks and the insurgency would not have seen such an increase in organizational maturity.*

Such a simulation would need to account for the various perspectives involved to determine whether other groups would react unfavorably to the Sunnis receiving stipends.

Gaia directs the developer to identify the relevant organizations in the domain. Table 5 shows the organizations that played major roles. Those organizations have been formed into the organizational model for the Iraqi case study.

**TABLE 5. GAIA’S ORGANIZATIONAL MODEL**

<b>General Population.</b> Generally united in the desire to drive the U.S. occupation out of Iraq and to take control of the opportunity that is shaping the future of Iraq.
<b>Sunni Arabs.</b> Privileged minority in Iraq’s population that failed to setup a representative government and found themselves without the wealth or identity they once held in the old regime.
<b>Shi’a Arabs.</b> Historically represented as the disadvantaged and oppressed population. They are disliked by the Sunnis.
<b>Former Ba’thist Regime.</b> Appointed by Saddam to government and civil positions and found themselves turned out of their positions when Saddam was overthrown.
<b>U.S. Stabilization Forces.</b> United in their effort to bring Iraq into a stable state and to establish the representative government necessary so that extremism cannot thrive.

The organizational model is followed by the environmental model, which indicates what data should be read by the agent system. The environmental model for the Iraqi case study was compiled from the Initiative on Security and Globalization Effects’ MPICE Framework of indicators and metrics for conflict transformation and stabilization<sup>1</sup>. The societal cleavage metrics translate into environmental sensors in the Gaia environmental model (see Table 6).

**TABLE 6. ENVIRONMENTAL MODEL SENSOR EXAMPLE**

SOCIETAL CLEAVAGES		
reads	hateAttacks	Incidence of hate crimes and attacks on symbols of group identity.
	inclinationTo Violence	Group acceptance of exclusionary social paradigms, readiness to use violence to achieve socio-political ends, including killing of noncombatants/innocent civilians.

The role model identifies the basic skills associated with each role. It is iterated throughout the rest of the design process. The roles that were identified included the occupied population, the insurgent, the old regime, the returning exiles, and the occupier. Table 7 illustrates the definition of the insurgent role.

**TABLE 7. GAIA ROLE DEFINITION EXAMPLE**

Role Schema:	Insurgent
<b>Description:</b>	This role is assumed by native Iraqis who see the fall of the old regime as a turning point for Iraq and want to see Iraq controlled by Iraqis, not the U.S.
<b>Protocols and Activities:</b>	ImpactConsentForThePeaceProcess DisruptSafetyOfElectionProcess TerrorizeElectionParticipants
<b>Permissions:</b>	Access to all reads variables. Access to all changes variables.
<b>Liveness Responsibilities:</b>	(DisruptSafetyOfElectionProcess cooperationAccepted < tipping point
<b>Safety Responsibilities:</b>	OLD_REGIME(INSURGENT[i]) INSURGENT(DisruptSafetyOfElectionProcess(OCCUPIED)) INSURGENT(TerrorizeElectionParticipants(OCCUPIED))

The interaction model defines the details for each protocol named in the role model. Table 8 defines the protocol the Old Regime and the Occupied Population roles would use to influence the peace process.

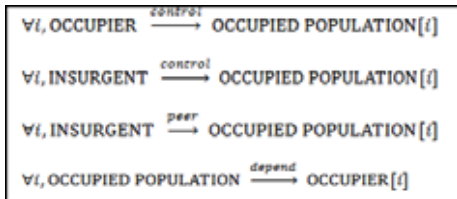
**TABLE 8. GAIA PROTOCOL DEFINITION EXAMPLE**

Protocol Name	ImpactConsentForThePeaceProcess
<b>Initiator</b>	Old Regime
<b>Partner</b>	Occupied Population
<b>Inputs</b>	securityStability basicNeedsStability diversityAcceptance
<b>Outputs</b>	convergence hateAttacks inclinationForViolence extremism

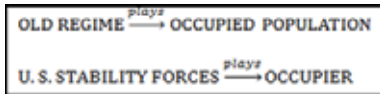
<sup>1</sup> Available at <http://www.sagecenter.net/search/node/MPICE>.

<b>Description</b>	This protocol evaluates the Old Regime's alternative approaches to disrupting the peace process. Its sensors into the environment are measures of the displaced population returning to Iraq to participate in the peace process. Its effect on the environment is expressed as intolerance for the peace process.
--------------------	--

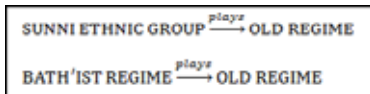
Once the protocols are defined, Gaia recommends analyzing control relationships to begin identifying the organizational structure:



Once the control relationships are identified, the developer can begin partitioning the agents. First, we capture the role of the old regime and the returning exiles as parts of the occupied population and the U.S. Forces as the occupier:



Next, we map the ethnic groups that make up the majority in the old regime and the returning exiles:



Finally, we identify the groups that have reason to join the insurgents.



Each of the artifacts above is analyzed to produce the services model. An example service definition is shown in Table 9.

**TABLE 9. GAIA SERVICE MODEL EXAMPLE.**

<b>Service:</b>	<b>Communicate Consent For The Peace Process</b>
<b>Inputs:</b>	qualityOfLife; hateAttacks
<b>Outputs:</b>	hopeForReform
<b>Preconditions:</b>	elections have been organized
<b>Postconditions:</b>	elected officials in place

#### IV: INTERPRETATION OF RESULTS

Gaia appears to be married to rules-based implementations early in the analysis in the same way that TROPOS and Prometheus are married to BDI implementations early on. Overall Gaia is less methodical about deriving the internal reasoning structure than it is in analyzing the organizations, roles, and agent partitioning. Gaia's strength is in the societal level analysis of the system, as opposed to the internal reasoning level analysis.

The findings of this research suggest that the AOSE methodologies are not necessarily competing approaches toward the design approach, but instead represent differing perspectives into several design approaches. Gaia does a good job of analyzing organizations and the roles that support those organizations. But once the Gaia process starts analyzing the rules that guide the behavior of each role, it is much less methodical than it was in the role analysis. Once Gaia analyzes the societal organization of agents, the internal reasoning system of an agent needs to be analyzed in a way that is independent of a particular architecture; the analysis should determine what type of architecture would be best suited to the system. Gaia, TROPOS, and Prometheus were all found to be biased toward a particular implementation in their early analyses.

Gaia seems a logical front end to all the other methodologies if some modifications and extensions were made:

- Role analysis should be carried out within the scope of the null hypothesis, control variables, and experimental variables.
- The organizational patterns literature needs to be summarized as a set of templates to guide the designer through which organizational pattern is best.
- The expression and classification of rules should be dropped in the early analysis and attention focused on defining the end state of interactions between agents.
- Rules should be established as part of the interaction model to express sequences without regard for what type of rule they are.
- AUML should be integrated into Gaia's design phase to take advantage of layered decomposition of activities and protocols.

#### V. RECOMMENDATIONS FOR FUTURE WORK

The early analysis phase in all the methodologies should be free of any conceptual mapping onto an operational construct (like a rule, a belief, desired state (goal), or intention). The end product of the analysis phase should be scoped definitions of the roles at play in the domain and a conceptual representation of the interactions between agents at the macro-level and the micro-level.

Our case study highlighted that group attitudes are not easily computable in BDI architectures. In reality, groups are made up of individual belief systems that coalesce into a shared belief system. When a BDI representation explicitly decomposes an individual attitude (i.e., goals, beliefs,

capabilities, actions, etc.), it is hard to compute the *group's* belief structure dynamically as individuals enter and exit the group. Rules-based expert systems are better at this because they are based on numerical computations instead of semantic nodes.

However, the drawback of rules-based systems is that they rely solely on numerical approximations of attitudes (such as an *anger* attribute scaled from 0 to 100) and therefore have no deliberate evaluation of alternatives the way BDI models do. Rule-based systems rely on probabilities that an event will happen or the degree that an entity possesses a particular attribute. The assessment of these attributes is often times highly subjective. Therefore, we assert the following:

1. When the interactions between agents take place at the societal level, and those interactions are constrained by a computable summation of *group* attitudes, the most computable implementation is the rules-based expert system. The BDI implementation would be too semantically expressive to compute.

2. When the interactions between agents take place at the micro- (individual to individual) level (that is, the interactions focus on an individual's belief system), the most deliberative implementation is the BDI implementation. An expert system may skim over important nuances of decision-making in the domain because the decisions are driven by probability equations, not a deliberate evaluation of alternatives in a particular context.

The next evolution of methodologies should focus on divorcing themselves from a particular implementation and providing design patterns that help a designer evaluate when a particular modeling technique or algorithm is appropriate.

Finally, we propose that the agent methodologies address criteria that are specific to causal social simulations:

- **Guidance on scoping the social experiment.** We recommend that the AOSE methodologies integrate an analysis step before the role analysis that scopes the experiment in terms of a null hypothesis, control and experimental variables, and provide some guidance on eliminating bias in the agent simulation. This is an important first step that constrains how many roles need to be analyzed.

- **Representation of uncertainty.** Many agent implementations need a way to interpret incomplete or uncertain information. Many agent implementations are made up of fuzzy logic nets or Bayesian networks, yet the methodologies fail to explain how to decompose the fuzzy logic.

- **Group belief system formation.** The methodologies need to account for complex systems that combine individual measures into a cohesive, aggregate belief system that represents a group belief system.

- **Computability.** The methodologies need to compute collective attitudes in such a way that system constraints and rules can be evaluated in a computable way.

## References

- [1] Schmidt, B. (2001). What are agents and what are they for? In N.J. Saam & B. Schmidt (Eds.), Cooperative Agents: Applications in the Social Sciences (pp. 5-20). Springer Publishing.
- [2] Antoniou, G. & van Harmelen, F. (2008). A Semantic Web Primer. Cambridge, Massachusetts: The MIT Press.
- [3] Hashim, H.S. (2006). Insurgency and Counter-Insurgency in Iraq. Ithaca, NY: Cornell University.
- [4] Wooldridge, M. & Jennings, N. (1995). Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 10(2), 115-152.
- [5] Wooldridge, M., Jennings, N., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents Multi-Agent Systems, 3, 3, (pp. 285-312).
- [6] Zambonelli, F., Jennings, N., and Wooldridge, M. (2003) Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM). Volume 12, Issue 3. Pp. 317-370.
- [7] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. Proceedings of Autonomous Agents and Multi-Agent Systems, (pp. 203-236). Kluwer Academic Publishers.
- [8] Padgham, L. & Winikoff, M. (2004). Developing Intelligent Agent Systems. West Sussex, England: John Wiley & Sons Ltd. [9] Odell, J., Van Dyke Parunak, H., & Bock, C. (2001). Representing agent interaction protocols in UML. In Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering. Lecture Notes in Computer Science, vol. 1957. (pp. 121-140). New York: Springer-Verlag.
- [10] Bauer, B., Muller, J., & Odell, J. (2001). Agent UML: A formalism for specifying multiagent software systems. Int'l Journal of Software Engineering and Knowledge Engineering, 11, (3). (pp. 207-230).
- [11] Sturm, A. & Shehory, O. (2003). A Framework for Evaluating Agent-oriented Methodologies. In Proceedings of the 5<sup>th</sup> Int'l Bi-Conf. Wkshp on Agent-Oriented Info Sys (AOIS), Springer LNCS 3030.
- [12] Lin, C., Kavi, K., Sheldon, F., Daley, K., & Abercrombie, R. (2007). A Methodology to Evaluate Agent Oriented Software Engineering Techniques. In Proceedings of the 40<sup>th</sup> Hawaii International Conference on System Sciences. (pp. 1-10).
- [13] Rao, A. & Georgeff, M. (1992). An abstract architecture for rational agents. In C. Rich, W. Swartout, & B. Nebel (Eds.), Proceedings of the 3<sup>rd</sup> International Conference on Principles of Knowledge Representation and Reasoning, (pp. 439-449). Cambridge: Morgan Kaufmann Publishers.

# REALIZATION OF SEMANTIC SEARCH USING CONCEPT LEARNING AND DOCUMENT ANNOTATION AGENTS

Behrouz H. Far<sup>1</sup>

Cheng Zhong<sup>1</sup>

Zilan (Nancy) Yang<sup>1</sup>

Mohsen Afsharchi<sup>2</sup>

<sup>1</sup>Department of Electrical and Computer Engineering, University of Calgary, Canada  
{far, czhong, zyan}@ucalgary.ca

<sup>2</sup>Department of Electrical and Computer Engineering, University of Zanjan, Iran  
afsharchim@iasbs.ac.ir

## ABSTRACT

*Currently, search systems are based on commitment to a common ontology. In the real world, it is preferred to enable Web repositories to exchange information freely while keeping their own ontology. This helps contents providers to represent information independently in the repositories at the expense of bringing complexity to the communication and negotiation. To solve the communication complexity problem we present (1) a method for semantic search supported by ontological concept learning, and (2) a prototype multi-agent system that can handle semantic search while encapsulating complexity of such process from the users. The method introduces a spiral search process and a layered structure of semantic interoperability. Agents, which conduct semantic search on behalf of users, deploy ontologies to organize documents in their corresponding repositories. Through a detailed experiment we will show that agents can improve their search capability by learning new concepts from each other, and consequently, provide better search results to the users.*

**Index Terms** — multi-agent system, semantic search, ontology, concept learning, interoperability, annotation.

## 1. INTRODUCTION

Current popular search engines are mainly divided into three common categories: horizontal, vertical and combination search engine. Horizontal search features keyword-based indexing and minimal natural language processing. Users need to evaluate the search results for obtaining desired documents. Vertical search indexes content specialized by location, topic, etc., typically tailored to users' preferences. Instead of returning thousands of documents, vertical search engines deliver more relevant results matched with the users' local needs. In 2007, Google introduced the "Universal Search" system that replaced some of search results with blended listings that come from vertical sources, such as news, video, images, etc. The blended search engine requires changes like re-categorizing,

reorganizing, and/or refining content of documents by grouping them by some attributes. This type of search engine typically works with a predefined ontology.

In contrast with the traditional keyword search technology which depends on the occurrence of words in documents, semantic search denotes one or more concepts in the context of other concepts. Understanding the denotation of concepts can help retrieval part of search engine understand the context of search, the activity the users is trying to perform, thus drive expectations on the categories of documents [5]. The essence of semantic search is *semantic interoperability* towards denotation part in the search phrase. Nowadays, general denotation procedures are realized depending on ontology-oriented means, and ontologies adopted are usually evolved and maintained in a distributed way. Thus, multiplicity of ontologies raises the issue of integration and may lead to ineffective communication among peers involved in a semantic search.

Multi-agent systems (MAS) research offers some solutions for the semantic interoperability. Recently, the idea of having agents *learn* concepts from peers has been suggested as a solution. For example, the work in [7] suggests a method for learning a language and the work in [10] has focused on interactions between two agents to learn a single concept. We have already presented a method for agents to learn concepts from several peers [1, 2] and a method for verification of the learnt concepts [4].

The goal of this research is to devise a process, a model and a prototype multi-agent system (MAS) for semantic search that features concept-learning and semantic interoperability. Research overview and MAS system design will be explained in Sections 2 and 3. A detailed experiment to verify usefulness of the prototype system is provided in Section 4 followed by conclusions in Section 5.

## 2. RESEARCH OVERVIEW

The general research goals of semantic search using concept learning MAS involves: (1) algorithms for concept learning; (2) methods of concept learning verification; and (3) cooperative search engine and supporting MAS. In this

paper we focus on the third goal, by creating a MAS that supports semantic search by taking advantage of concept learning and verification. In order to achieve the goal, the followings objectives must be fulfilled:

- 1) Individual agents are capable of learning ontological concepts from several peer agents through the interaction with other agents and validating these concepts to better communicate and share information.
- 2) Semantic search engines are capable of dynamically annotating the data repositories.
- 3) An integrated method or mechanism is required to support and facilitate the implementation of complex interactions among agents.

To achieve the first objective, ontological heterogeneity in MAS must be solved. This is directly related to the fact that any ontology of certain domain can potentially evolve independently. Therefore the only way for agents with diverse views of the world to understand each other is being able to understand each other's conceptualization of the domain, and then find common grounds among themselves. Previous works on agents' communication mostly assumed a complete common understanding of the concepts is used to represent a domain. However, it is now known that this may not be necessarily true. Even if having common conceptualization, still the agents are required to be aware that they have a common conceptualization using mechanisms such as social networking. This fact is summarized with the point that any conceptualization is invented based on its utilization [8]. Consequently, ontology learning solutions are gaining more popularity [9, 10].

To achieve the second objective (i.e. semantic search engine), more advanced than a typical query handling system, we have devised a spiral workflow process that incorporates both concept learning and semantic search (See Figure 1). On one hand, search engines should be capable of responding to the requests according to agreements with concept learning module. On the other hand, annotation procedures of search engine can be done on the fly based on the newly obtained concept instead of fixed predefined ontological concepts. This is a novel view exposing intrinsic relationship between concept learning and semantic search in a heterogeneous environment. In such environment, concept-learning and semantic search are treated equally as basic roles, involved in the process, which support each other to achieve their own goals by enriching the set of ontological concepts and reducing ambiguity of the search, respectively. Following the spiral process, concept-learning module and semantic search take actions alternately.

To achieve the third objective, the problem of integration and communication between agents raised by multiplicity of ontologies need to be solved. As the essence of semantic search is semantic interoperability among different agents towards denotation part contained in the search expression, semantic search is expected to be able to take advantages of concept learning to establish an integrated mechanism to help find common understandings of concepts, and based on

it, higher-level modalities of ontology may accomplish interoperations with respect to those denotations.

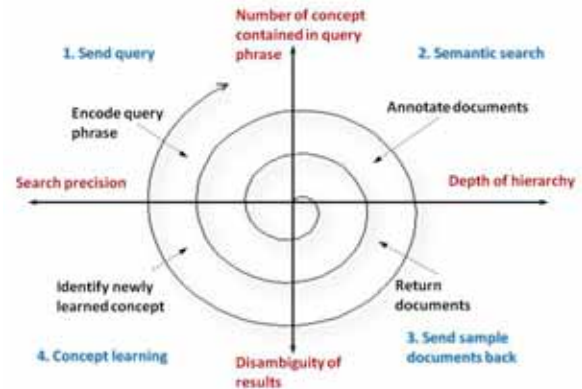


Figure 1. Spiral search and learning process

The work introducing the algorithm for agents to learn concepts from several peer agents (objective 1) has been presented in [1, 2, 12] and a method of verification of concept learning has been presented in [4]. Also the work regarding an initial implementation of a semantic search engine has been presented in [13].

### 3. SYSTEM DESIGN AND IMPLEMENTATION

Based on the semantic interoperability model [3], we have devised the layered semantic search architecture composed of encoding, lexical, syntactic, semantic and semiotic layers (see Figure 2). According to the definitions of functionalities of layers [13], in order to achieve the interoperations between peers, modeling semantics of concepts and use them in the semantic and semiotic layers need external "schema" (i.e. procedural knowledge), however, for the lexical layer, the declarative contents could solely accomplish modeling by referring concepts to some commonly-understood objects. Considering the fact that the concept learning module [1, 2] is built with a kind of declarative concept learning algorithm, i.e. concentrating on lexical layer, the current implementation of the prototype also has focus on the lexical layer.

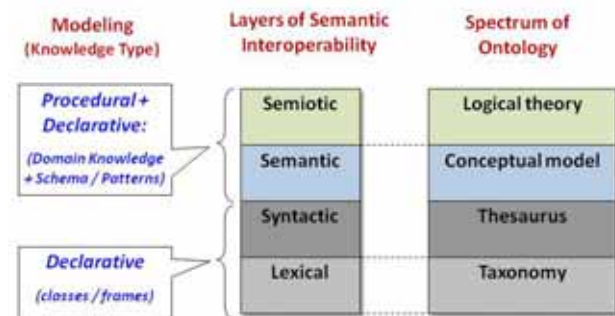


Figure 2. Scope of the prototype system

Furthermore, from the cognitive sciences perspective, lexical layer would be a basis of communication which ultimately leads to understandings of semantics, so that successfully achieving semantic interoperation would lay a solid foundation for other layers.

Figure 3 shows architecture for the prototype system. System’s functional blocks are briefly introduced below.

### 3.1. Document Annotator (DA)

The documents annotator is used for annotating “molecules”, combinations of keyword, on which some well-defined constraints are applied. Creating such annotation, especially dynamically creating annotations is a fundamental role, not only for concept learning, but also for semantic search involving newly learnt concept. Document Annotator is developed using IBM’s UIMA (Unstructured Information Management Architecture) [6]. Annotator implements actions *SelectBestConcept*, *SelectPosEx* and *CreateNegEx* according to the UIMA annotation scheme.

### 3.2. Concept Learner (CL)

The concept learner is responsible for implementing action *Learn* which takes training documents as input and output concept classifier. Also it offers function to do action *Integrate*.

### 3.3. Communication Engine (CE)

Communication Agent implements actions *QueryConcept* and *ReplyQuery* which facilitate agent communication.

### 3.4. Personal Assistant Agent (PAA)

Currently, there are two types of PAAs – Training Application (TA) and Semantic Search Application (SSA).

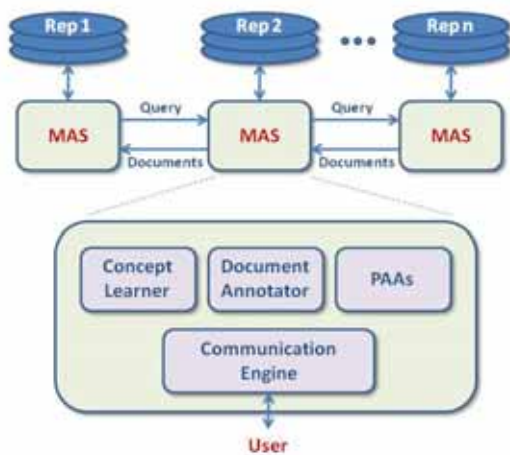


Figure 3. Prototype system and MAS components

GAIA analysis and design methodology [11] is used to design the MAS that implement the above mentioned functions. The MAS associated with each repository is composed of 4 types of agents – Concept Learner,

Document Annotator, PAAs and Communication Engine, as shown in Figure 3.

## 4. EXPERIMENTAION AND EVALUATION

We have designed three experiments using the developed prototype system to observe how the evolution of search results is influenced by concept learning and semantic search and compare them with traditional search. In Experiment 1, a series of traditional queries will be processed by the agents residing on data repositories (Figure 3) in order to observe behaviors of a traditional search and to set benchmarks for comparing with the results of other experiments. Experiment 2 is designed to observe the concept learning stage of the spiral search process. Before sending queries, a new concept is supposed to be identified through interactions between Concept Learner (CL) and Document Annotator (DA) agents, and using the attributes of the new concept, the initial repositories are re-structured to be hierarchical repositories. Experiment 3 is conducted using the hierarchical data repositories of Experiment 2. It represents the stage of semantic search of the spiral process. The queries are processed after the annotation process in which annotators initiatively annotate data repositories they are handling with the same type system which is designed to filter documents.

The disambiguation of search results is measured by a metrics named ROD (Ratio of Disambiguation) which represents the precision of query results.

$$ROD = \frac{Pos}{Pos + Neg} \times 100\%$$

- Pos: number of positive documents. The contents of a positive document meet the query conditions.
- Neg: number of negative documents. A negative document is a false positive document.

The positive or negative is determined by a human expert.

### 4.1. Test data set

The test data set consists of files describing course syllabi in Computer Science offered by three major universities. A course syllabus file normally contains a course identifier, a course description and the prerequisites of a course. The University of Michigan organizes Computer Science (EECS) as an engineering discipline and as a joint program with electrical engineering; the University of Washington considers Computer Science (CSE) as an engineering discipline but independent from electrical engineering and as a joint program with computer engineering; in Cornell University Computer Science (CS) is a pure science program in the science faculty. The three universities together offer 279 courses in electrical engineering and/or computer science, excluding some courses such as seminar course. We set up three data repositories, one for each university, with each repository having a MAS (Figure 3) to handle it. The  $Ag_C$ ,  $Ag_W$  and  $Ag_M$

stand for the MAS handling Cornell University, University of Washington, and University of Michigan, respectively.

#### 4.2. Experiment setting

The search goal is to find all courses in *programming languages* from the three data repositories. Query phrases utilized for the three experiments are constructed with five keywords which are related to the search goal. There are five query phrases are listed in Table 1.

Table 1. Query phrases

Query ID	Feature Content
F1	Language
F2	Language, Program
F3	Language, Program, Computer
F4	Language, Program, Computer, Science
F5	Language, Program, Computer, Science, Software

#### 4.3. Experiment 1: Traditional search

Traditional search is conducted in Experiment 1. The result is recorded in Table 2, and visualized in Figure 4.

Table 2. Results summary: Experiment 1

	$Ag_C$			$Ag_M$			$Ag_W$		
	Pos.	Neg.	%	Pos.	Neg.	%	Pos.	Neg.	%
F1	4	1	80	4	16	20	4	15	21
F2	6	2	75	8	30	21	6	28	18
F3	6	5	55	8	55	13	6	50	11
F4	6	7	46	8	56	13	6	51	11
F5	6	7	46	8	62	13	6	55	11

Examining the record of Experiment 1, we can find that the ratio of disambiguation of  $Ag_C$  is much higher than the  $Ag_M$  and  $Ag_W$ . We think that this is caused by different composition of data repositories.  $Ag_C$  actually holds courses of pure computer science, whereas  $Ag_M$  and  $Ag_W$  manage courses with composition of both computer science and electrical engineering.

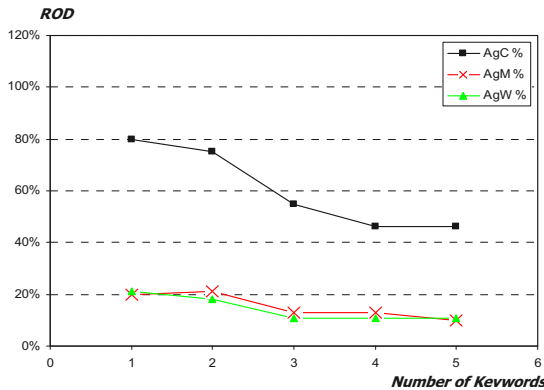


Figure 4. Visualization of results: Experiment 1

In addition, it is worth to mention that:

- 1) The more terms added to each query, the more documents were retrieved, regardless of whether the documents were positive or negative.
- 2) Ratios of disambiguation were not guaranteed to be improved with terms added to the query. In this case, it caused the ratios to get worse by adding more terms.
- 3) All three data repositories were isolated so the number of positive documents was definite. The queries with feature F2 obtained all positive documents in the repositories. After that, no other positive document could be found and the search noise made results worse.

From this experiment we can conclude that the composition of data repository influences the search results, confirming that the expected results are significantly correlated with the data repository.

#### 4.4. Experiment 2: Search with Concept Learner

Experiment 2 focuses on examining the behavior of queries, when the Concept Learner has been introduced. The algorithm built into the Concept Learner takes the same data repositories as in Experiment 1 to identify a new concept, *Computer Science*, and then using it to identify all its subcategories. Using the learnt concept, the data repositories are reorganized for the subcategories of *Computer Science* [1]. One subcategory, the *programming languages*, is directly adopted to annotate data repositories when annotating action is performed.

Through concept learning, the initial flat data repositories were restructured to a two-level hierarchy. We repeated the queries as in Experiment 1 on these structured data repositories. The queries were no longer traditional because at this point any query would have been assumed by the search engine to be a query for all courses of *Computer Science*. In practice, new concept (in this case *Computer Science*) will be involved in each query feature to semantically describe it. The results of Experiment 2 are recorded in Table 3 and visualized in Figure 5.

Table 3. Results summary: Experiment 2

	$Ag_C$			$Ag_M$			$Ag_W$		
	Pos.	Neg.	%	Pos.	Neg.	%	Pos.	Neg.	%
F1	4	0	100	4	4	50	4	6	40
F2	6	0	100	6	10	38	6	13	32
F3	6	3	67	6	13	32	6	19	24
F4	6	4	60	6	13	32	6	19	24
F5	6	4	60	6	20	23	6	19	24

For Experiment 2 we can conclude that:

1. RODs have been improved for all the three repositories and for all the queries. Intuitively, as shown in Figure 5 all the lines representing trends of change of RODs have shifted up significantly.
2. Variations of ROD are still following the same trend as in Experiment 1 (i.e. with the terms added to query, the RODs are decreasing).



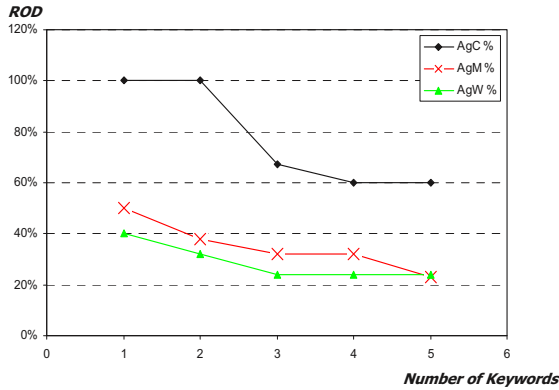


Figure 5. Visualization of results: Experiment 2

The reason for RODs to be different is due to the differences of composition among the data repositories. In the data repositories mixing courses of both disciplines Computer Science and Electrical Engineering such as  $Ag_M$  and  $Ag_W$ , irrelevant courses (e.g. electrical engineering related courses) were eliminated more effectively than that of pure data repository as those of  $Ag_C$ , only holding courses of computer science.

#### 4.5. Experiment 3: Search with document annotation

From the results of Experiment 2, we could conclude that through applying concept learner, search performance improves. However, the trends of ROD remained the same as in Experiment 1.

In this experiment, we apply the Document Annotator (DA) agent to semantically determine if a document is about the searched concept or not, and to see how search performance would be influenced.

Experiment 3 was carried out based on the refined data repositories in Experiment 2. At the beginning of the Experiment 3, each data repository was annotated with the same UIMA [6] type system (i.e. kind of concept hierarchy). An aggregate annotator was established consisting of a series of primitive annotators for annotating terms including *language*, *program*, *C*, *C++*, and *Java*. As all the documents to be scanned and relocated, were already under computer science, we were able to replace those non-domain specific terms (computer, software, and science) with those specific terms of the domain computer science (*C*, *C++*, and *Java*). The following expression illustrates a typical annotation logic of the aggregate annotator:

<Language + Program + [C|C++|JAVA] →  
Computer Programming Course>

This can be interpreted as: “if a three-concept entity created through some logic built in the annotator has been found in the document, then this document is a target course, i.e. *computer programming language course*.”

Once the annotation process was completed, the corresponding alteration to the current data repositories was made. Documents that had not been annotated successfully were removed from the sub-directory dedicated to computer

programming language course description. Hence, the ratios of positive documents were raised and the noise that was brought in by adding terms to the query was reduced.

Through the Experiment 2 and the annotation process of the Experiment 3, data repositories were structured with two levels: applying concept learner in Experiment 2 and annotation in Experiment 3. Then we continued to process queries with the same set of features on these restructured repositories. The results of Experiment 3 are listed in Table 4, with visualization in Figure 6.

Table 4. Results Summary: Experiment 3

	$Ag_C$			$Ag_M$			$Ag_W$		
	Pos.	Neg.	%	Pos.	Neg.	%	Pos.	Neg.	%
F1	4	0	100	4	4	50	4	6	40
F2	6	0	100	6	10	38	6	13	32
F3	6	0	100	6	10	38	6	13	32
F4	6	0	100	6	10	38	6	13	32
F5	6	0	100	6	10	38	6	13	32

Compared to Experiment 2, the RODs for the first two queries (i.e. F1 and F2) remained the same as Experiment 2, but the RODs of the rest of queries (with features F3-F5) showed improvement. The reason that the trend lines are more or less horizontal is that adding terms to queries no longer brings noises as in the previous experiments because the sources of noise (i.e., irrelevant documents) had already been removed.

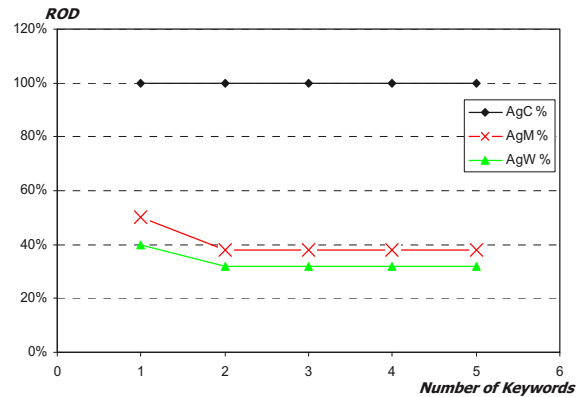


Figure 6. Visualization of results: Experiment 3

#### 4.6. Experiment evaluation and summary

Contribution to the improvement of search results made by both Concept Learner (CL) and Document Annotator (DA) agents is the main concern. In order to evaluate contributions made by concept learning and annotation, the percentages of increment of ROD for each query and their average, contributed by concept learning and annotation, are computed respectively. The results are listed in Table 5. As the  $\Delta R1$  and  $\Delta R2$  in Table 5 indicate:

- The average rate of increase of ROD achieved through concept learning on  $Ag_C$  (28%) is much less than those on  $Ag_M$  (121%) and  $Ag_W$  (121%).

- The average rate of increase of ROD achieved through annotation on  $\mathcal{A}_{g_C}$  (%37) is larger than those on  $\mathcal{A}_{g_M}$  (21%) and  $\mathcal{A}_{g_W}$  (20%).
- Both concept learning and annotation made almost identical contributions on data repositories  $\mathcal{A}_{g_M}$  and  $\mathcal{A}_{g_W}$ . The reason is that  $\mathcal{A}_{g_M}$  and  $\mathcal{A}_{g_W}$  are mixed data repositories, therefore concept learning had more significant effect on them than on  $\mathcal{A}_{g_C}$ . However, later in the spiral process, composition of the three data repositories becomes increasingly similar, and consequently, annotation affected the results similarly.

**Table 5. Comparison of results**

Data Repository	R1(%)	R2(%)	R3(%)	$\Delta R1(\%)$	$\Delta R2(\%)$
$\mathcal{A}_{g_C}$	80	100	100	25	0
	75	100	100	33	0
	55	67	100	22	49
	46	60	100	30	67
	46	60	100	30	67
	<b>Avg.</b>			<b>28</b>	<b>37</b>
$\mathcal{A}_{g_M}$	20	50	50	150	0
	21	38	38	85	0
	13	32	38	146	19
	13	32	38	146	19
	13	23	38	77	65
	<b>Avg.</b>			<b>121</b>	<b>21</b>
$\mathcal{A}_{g_W}$	21	40	40	90	0
	18	32	32	78	0
	11	24	32	118	33
	11	24	32	118	33
	11	24	32	118	33
	<b>Avg.</b>			<b>121</b>	<b>20</b>

R1: Values of ROD of Experiment 1; R2: Values of ROD of Experiment 2; R3: Values of ROD of Experiment 3;  $\Delta R1$ :  $(R2-R1)/R1$ ;  $\Delta R2$ :  $(R3-R2)/R2$

Therefore we could conclude that using either concept learning or annotation in isolation would not necessarily lead to a noticeable search improvement but the sequential use of them in the generative/spiral process can potentially lead to a major improvement.

## 5. CONCLUSIONS

In this paper we presented a method and a prototype MAS for semantic search-learning. This method is based on the architecture of layered semantic interoperability. The central procedure is composed of dynamical document annotation and concept learning mechanisms to solve the problem of semantic heterogeneity in distributed information management with minimum overhead and no need to commit to a common ontology. A detailed experiment was conducted on three data repositories with different ontologies within a specific domain. The experiments were focused on two major parts of the spiral process of semantic search and concept learning. The findings were:

1. When contents of data repositories are relevant to the query keywords, the composition of the data repositories influences the search results. Adding keywords to the query is not helpful for disambiguating the results.
2. Both Concept Learner (CL) and Document Annotator (DA) agents play significant role in refining the compositions of data repositories in different ways: CL achieves the improvement through reconciling the conflicts of concept between the holders of data repositories, guided by attributes of the newly learned concept. DA, on the other hand, works on its own data repository by applying individual annotation algorithms to restructure the contents.

Future work includes implementation of mechanisms for finding peers through social networking which will lead to an open MAS for semantic search.

## REFERENCES

- [1] M. Afsharchi, B.H. Far and J. Denzinger, "Enhancing Communication with Groups of Agents Using Learnt Non-unanimous Ontology Concepts," Journal of Web Intelligence and Agent Systems, vol. 3, no. 1-3, pp. 1-16, 2007.
- [2] M. Afsharchi, B.H. Far, J. Denzinger, "Ontology Guided Learning to Improve Communication among Groups of Agents," Proc. AAMAS'06, pp. 923-930, 2006.
- [3] J. Euzenat, "Towards a principled approach to semantic interoperability," A. Gomez-Perez et al (eds.) IJCAI'2001 Workshop on Ontologies and Info Sharing, Seattle, 2001.
- [4] B.H. Far, A.H. Elamy, N. Houari and M. Afsharchi, "Adjudicator: A Statistical Approach for Learning Ontology Concepts from Peer Agents," The 19<sup>th</sup> Int. Conf. on Software Engineering and Knowledge Engineering (SEKE 07), 2007.
- [5] R. Guha, R. McCool, E. Miller, "Using the semantic web: Semantic search," Proceedings of the WWW'03, 2003.
- [6] IBM, "Unstructured Information Management Architecture (UIMA)", [http://domino.research.ibm.com/comm/research\\_projects.nsf/pages/uima.index.html](http://domino.research.ibm.com/comm/research_projects.nsf/pages/uima.index.html), 2007.
- [7] K.C. Jim, C.L. Giles, "Talking Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem," *Artificial Life* 6(3), 2000, pp. 237-254.
- [8] M.R. Genesereth, and N.J. Nilson, "Logical Foundation of Artificial Intelligence," Morgan Kauffman Publishers. Inc. Palo Alto. CA, 1987.
- [9] L. Steels, "The origins of ontologies and communication conventions in multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, 1(2):169-194, 1998.
- [10] A.B. Williams, "Learning to Share Meaning in a Multi Agent System, *Autonomous Agents and Multi Agent Systems* 8(2)," pp. 165-193, 2004.
- [11] N. Wooldridge, and D. Kinny, "The GAIA methodology for Agent-Oriented Analysis and Design," 2000.
- [12] Y. Zilan, C. Zhong, B.H. Far, "A Practical Ontology-Based Concept Learning in MAS," Proc. IEEE CCECE'08, pp. 335-338, 2008.
- [13] C. Zhong, Y. Zilan, M. Afsharchi, B.H. Far, "Ontology Learning Supported Semantic Search Using Cooperative Agents," The 20<sup>th</sup> Int. Conf. on Software Engineering and Knowledge Engineering (SEKE 08), pp. 123-128, 2008.

# Agent-based Simulation Model for the Evolution Process of Open Source Software<sup>1</sup>

Taemin Seo and Heesang Lee\*

Dept. of Systems Management Eng., Sungkyunkwan University, Korea

## Abstract

The *Open Source Software* (OSS) system is a type of software that is developed and evolved through voluntary developers and users. We illuminated the relationship among developers, users, and OSS with the evolution process of OSS in this article. We analyzed prior literature of OSS to determine the roles of developers and users who participate in OSS projects and described the evolution process of OSS using an agent-based simulation model. We performed various computer simulations to analyze the relationship among developers, users and the results of OSS. We also studied factors that affect the evolution process of OSS.

## 1. Introduction.

*Open Source Software* (OSS) is software that has been developed and improved by voluntary developers who share the source codes with other people for continuous evolution of the program. The collaboration experiments of many developers, the GNU project and free software foundation, and the users who want to choose freely among many software lead to this open environment for source code. The OSS growth demonstrates a phenomenon that increases at rapid rates, unlike proprietary software.

The purpose of this article is to illuminate the relationship with the evolution process of OSS among various agents by observing and/or researching of certain agents in the evolution process. The simulation model presented here describes the linear and fast growth of OSS by combining various factors, and provides a variety of possible alternative experiments along with changes of parameters

Based on prior literature and various OSS project materials, we determined that the factors which affect the evolution process of OSS projects are the roles of two agents, developers and users, and their changes during the evolution process. We determined the essential factors in the evolution process of OSS through various simulation

experiments using a multi-agent based simulation software for computer simulation.

This article is divided into five sections. Section II describes various related studies of the OSS phenomenon and evolution cases which were used in our research models. Section III presents our research model for the evolution process of OSS by developers and users. Section IV introduces our agent-based simulation model using a multi-agent based simulation program. Section V discusses our analysis of the factors that affect the evolution process of OSS by changing various parameters. Section VI concludes the article and mentions topics for further research.

## 2. Related Research Works

In a study of the motivations for participating in OSS [1], Hars and Ou classified the participant motivation of developers as *internal factors* or *external rewards*. The internal factors arise from participants' personal hobbies and preferences. The internal factors are reaped from working to increase the welfare of other people. The internal factors also include community identification, motivation by the feeling of competence, satisfaction and fulfillment that arises from writing a program, and altruism, which is a variant of intrinsic motivation. The external rewards include obtaining direct or indirect rewards by increasing their marketability and skill base or by selling related products and services. The external rewards also include future rewards such as revenues from related products and services, human capital, peer recognition and personal needs.

In a study of the OSS community [2], Xu and Madey divided those people participating in the OSS into two groups: the user group and the developer group. The user group includes both the *passive user* free from direct contribution and the *active user* who reports software bugs and new needs. The developer group is classified into four classes: peripheral developer, central developer, core developer, and project leader.

In a study of the growth and evolution of OSS [3], Godfrey and Tu found that the OSS demonstrated super-linear growth versus the sub-linear growth of proprietary software. This phenomenon was explained by the fact that the number of developers who participated in the OSS was not limited.

In a study of agent-based simulation of open source

<sup>1</sup> This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (R01-2008-000-10500-0).

\* Corresponding author. Email: leehee@skku.edu

evolution [4], Smith, et al. regarded the agents as both developers and unfulfilled requirements. They presented an agent-based simulation model that included the complexity of software modules, which was a limiting factor in the evolution using the NetLogo program. The model also added the fitness of software and the motivation of developers to its requirements. The developers could create, modify and re-factor the modules that included fitness and complexity in the model. These results indicated that increasing the number of developers yielded better results in terms of the system size.

In a study of the role of core developers in OSS development [5], Long investigated the factors that affect the success or failure of OSS projects. Focusing on the role of core developers, he divided the developer group into the core developer group and the community group. He analyzed an established theoretical framework based on the organization theory and empirical data for 300 OSS projects. He found that the leadership and activity of core developers was an important factor in the development of OSS projects.

### 3. Research Models

#### 3.1. Basic Research Model

Our basic research model for the OSS evolution process is illustrated in Fig. 1. In this model, there were three components that had interdependent relations with each other: the *developer group* that represents the people who develop an OSS, the *user group* that represents the people who use the OSS, and the *OSS itself*. We conjecture that the present success of OSS cannot be achieved if one of the three components interacts poorly.

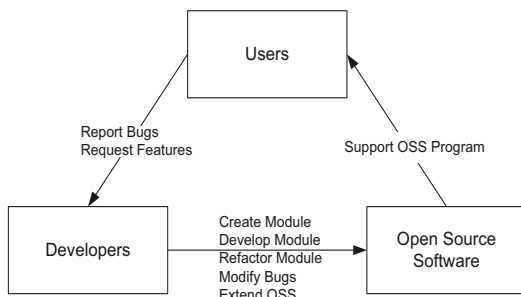


Fig. 1. Basic Research Model

More detailed interconnections among the three components are explained in the following sections.

#### 3.2. Role of Developers in the Proposed Model

We investigated the role of the developer group that participates in the OSS and its effect on the evolution of OSS projects. There have been several research studies performed to determine the reason for participation in OSS [1, 7, 8, 9, 10]. Some of the potential reasons include the

increased satisfaction through skill advancement and participation in personal hobbies and interests. Other potential reasons include participation to achieve future rewards or economical profit and to improve one's own social popularity and reputation through time, effort and experience. The motivation for participating in the OSS and the attraction of OSS, which are expressed in our model by the rate at which the OSS can attract participant's interests, are important factors that affect the evolution process of OSS.

The *core developer group*, which usually consists of 5 to 15 persons from the developer group for an OSS project, was also important for our model. The rate of increment of the OSS source codes increases rapidly when the core developer group is active in controlling the structure and direction of development [5, 11]. The core developer group definitely contributes to the progress of the OSS as they manage the CVS (Concurrent Version System) and present the goal and direction of the OSS to the developer group. Therefore, their role among developer groups and activity areas are important factors that affect the evolution process of OSS in our model.

A refactoring task is an activity that readjusts the structure of codes without changing the results. This task generally increases the readability and carries out maintenance while decreasing the complexity of the code, which is different from the task of removing software bugs and adding new features. Therefore, refactoring refers to the maintenance tasks that change and improve inner structures without affecting the output seen by the user group. As the projects progress in the early stage, the code becomes more complex since the OSS progresses around the developer group without a central control mechanism [6]. However, after progressing to a certain stage of evolution, the developer group improves the level of OSS while decreasing the complexity of codes through refactoring. Therefore, the task of refactoring is also an important factor that affects the evolution process of OSS.

#### 3.3. Role of Users in the Proposed Model

Within the user group, we assumed that there are a *passive user group* and an *active user group* [2]. The passive user group only downloaded and used the programs for personal needs, while the active user group reported software bugs and requested new requirements for the OSS in our model.

When the user group chooses the programs, the attraction of OSS is as important as the actions of the developer group since the user group wants programs that suit their needs. It is important to determine whether the programs are sufficient for the user group in order to indicate the desired features or needs. Therefore, the attraction of OSS is an important factor that affects the evolution process of OSS in our model. We also assumed that the rate of the active user

group, which affects the evolution process of OSS, was an important factor in our model since the active user group makes a direct contribution to the evolution process of OSS. The number of software bugs found by the active user group increases the amount of work for the developer group. The new requirements requested by the active user group necessitate OSS extension work. Therefore, the software bug detection and feature requirements are also important factors that affect the evolution process of OSS in our model.

Through these investigations, we developed our detailed research models that include the roles of developers and users and their effect on OSS. The detailed models are shown in Figs. 2 and 3.

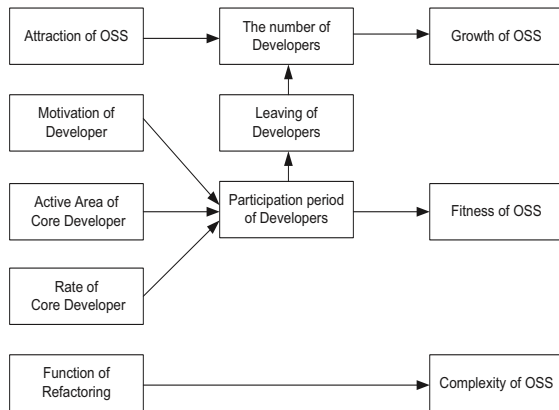


Fig. 2. Detailed research model for the role of developers

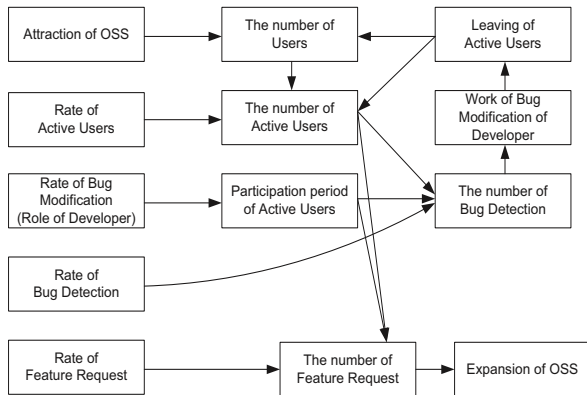


Fig. 3. Detailed research model for the role of users

In the next two sections, we examine the relationship among developers, users and the OSS through computer simulation based on these detailed models.

#### 4. Modeling for Relations and Roles

We attempt to understand and model the relationships and roles among developers, users and the OSS in this section. As we mentioned above, we focused on the analysis and derivation of factors that affect the evolution process of OSS. We executed the computer simulation using the AnyLogic program, which is a multi-method simulation

software that supports the most common simulation methodologies. It also includes a wide range of data analysis and business graphics objects such as bar charts, time plots and histograms. Once a computer simulation model is complete, we can use it to run various experiments by using various parameter settings [14]. The agents in our model were the OSS, the developer group and the user group. We based our model on the theory of the OSS evolution process presented by Smith et al. [4]. We assumed that the developing OSS had one project that consisted of several *modules*. Our model included an *initial development area* (2,500 modules) and an *expansion development area* (2,400 modules). The initial development area was assumed as an area that the developer group initially chose and the expansion development area was an area that would be developed for the expansion by the additional requirements of the user group. The OSS also had *attractions* that could affect participation in the developer group and the user group.

Each module had *fitness* and *complexity* in our model. The fitness is regarded as the length or the completion level of the software code, the file size and so forth. The complexity is regarded as the complicated level of each module. As the complexity of an OSS increases, the more difficult it is for the developer group to understand the software code. Therefore the complexity will negatively affect the increasing of the fitness.

The developer group consisted of the *core developer group* and the *general developer group* in our model. Both developer groups participated in the OSS by attraction to the OSS and developer recommendations, and seceded from the OSS because of the inner and outer motivation of each developer. The recommendation by the developer group does not appear in the early stages of the development, rather it appears when the number of developers is above a certain value. We assumed this in order to reflect the network effects.

A person in the developer group moved randomly around neighborhood modules in our model. At regular intervals, they chose one of the following behaviors:

1. If a developer's probability was below a certain value, they did nothing.
2. If a developer was on a module with software bugs, they modified the bugs.
3. If a developer was on a module that was not yet created, they created the module with low fitness and low complexity.
4. If a developer was on a module with a certain fitness and low complexity, they developed the module to increase its fitness and complexity.
5. If a developer was on a module with high fitness and high complexity, they re-factored the module to decrease its complexity.

The core developer group manages and develops general projects. We assumed that the core developer group's movement among modules was frequent and their level of code writing skills was high. On the other hand, the general developer group does not take direct responsibility for the project. Their movement among modules is not frequent and their level of code writing skills is relatively low. The core developer group also has an active area. If a person in the general developer group was not included inside the active area, they were regarded as being harmful to the mission and vision of the projects and their outer motivation was decreased in our model. A member of the core developer group may become bored and secede from the OSS when the fitness of the module was above their inner motivation in our model. A person in the general developer group seceded from the OSS when they reached the point described above or when their outer motivation equaled zero.

People in the user group reported bugs and requested the need for new areas or features. They jumped randomly around modules. At regular intervals, they chose one of the following behaviors:

1. If a user was inside the developing area, they found the existence of software bugs and reported the software bugs.
2. If a user was in the expansion area, they communicated new requirements to the developer group. If the number of user needs was above a certain value, the OSS was extended and evolved gradually.

The user group could develop grievances and secede from the OSS when the software bugs they found were not modified by the developer group.

When the simulation began, OSS started with a single core developer at the same time in our model. After a certain amount of time passed, the developers and users who were interested in participating in the OSS joined their respective groups. Our simulation model using the AnyLogic program is detailed in Fig. 4.

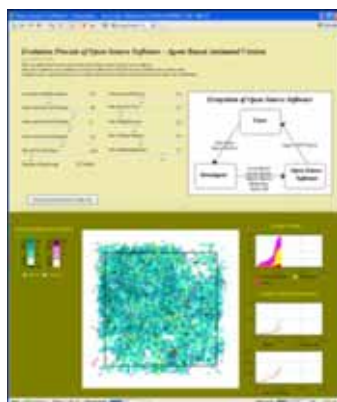


Fig. 4. Simulation model for the evolution process of OSS

## 5. Experiment Results

We executed the simulation of the OSS evolution process for 1 year. Its time unit was hours and its total time was 2,924 units. The parameter values used in this model are noted in Table 1.

TABLE I  
PARAMETER SETTING

Parameters	Value
Attraction of OSS for Developers	0.3
Inner Motivation of Developers	80
Outer Motivation of Developers	6
Active Area of Core Developers	10
Rate of Core Developers	0.05
Attraction rate of OSS for Users	0.4
Rate of Active Users	0.1
Rate of Bug Detection	0.3
Rate of Feature Request	0.4
Rate of Bug Modification	0.7
Function of Refactoring	true

Based on the parameter set, we executed ten computer simulation experiments and then could draw a graph from the data that included the average OSS growth rate of 10 experiments as time passes.

Fig. 5 shows the evolution process with a high growth rate of OSS. In Fig. 5, we assumed that the project started on January 1. The growth was slow in its early stage, but it grew rapidly after March. Then, as the project became more complete, its growth rate slowed again. Therefore, the variation throughout the project was close to the stretched S curve. The speed variations in this model are caused by an increase in the number of developers and users and their time of contribution throughout the given parameters. The project growth starts slowly due to the low number of developers and users participating in it. After a certain time, growth speeds up due to an increase in the number of developed modules as the number of developers and users attracted to the project increases and participation grows. When the project no longer evolves, the developers and users participating in the project start to lose interest in the

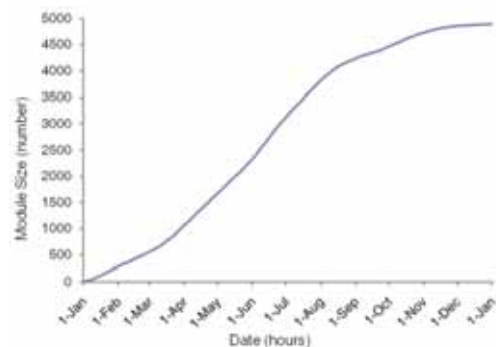


Fig. 5. Evolution process in high growth of OSS

development and secede from the project. Accordingly, its growth rate gradually slows.

### 5.1. Relation between OSS Attraction and OSS Growth

We examined how the attraction of OSS affected the OSS growth. First, the parameter value for the participation rate of the developer group changes from 15% to 35% in 5% increments and 10 simulation runs were executed for each case. Fig. 6 shows the difference in the growth of the OSS based on various participation rates of the developer group. This figure indicates that the 30% and 35% attraction rates for developers produce a higher growth rate of OSS. As developers are increasingly attracted to OSS, more developers will participate in the OSS. This causes the growth rate to increase because the development of modules occurs more often.

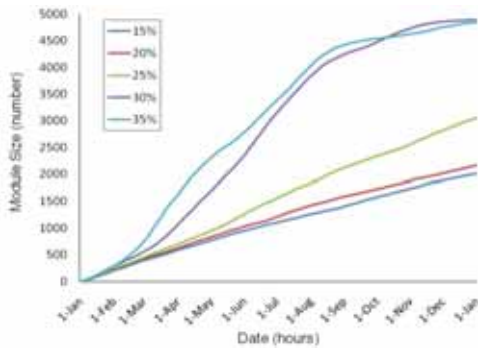


Fig. 6. Difference of OSS growth according to the change of attraction rate of developer group

Next, the parameter value for the participation rate of the user group changes from 15% to 35% in 5% increments and 10 simulation runs were executed for each case. Fig. 7 shows the growth difference according to the change in participation rates. It also shows that as users become more attracted to OSS, the growth rate for OSS increases. As users are increasingly attracted to OSS, more developers will participate in the OSS because of increasing user's new needs. Therefore, the growth rate is higher as the number of developer increases and the development area is extended for new requirements.

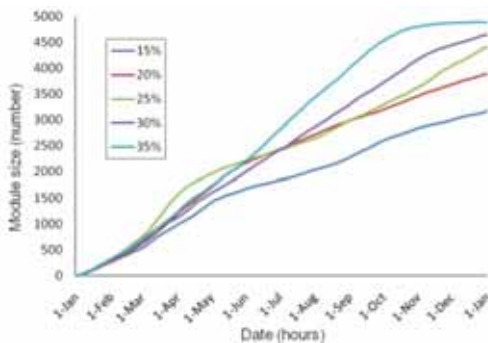


Fig. 7. Difference of OSS growth according to the change of attraction rate of user group

There is not much difference in the rate of growth in the early stages of the OSS, since the file size is less than 2500, the OSS is developed in the initial development area, and after the file size reaches 2500, it is developed in the expansion area according to the demand of users.

### 5.2. Developer's Motivation and OSS Growth/Fitness

We examined how the developer's motivation affects the growth and fitness of OSS. First, the parameter values for inner motivation changed from 50 to 80 by intervals of 10 and 10 simulation runs were executed for each case. Fig. 8 shows the difference in growth according to the change in the parameter value. Fig. 9 shows the difference in the average fitness and complexity according to the change in the parameter value.

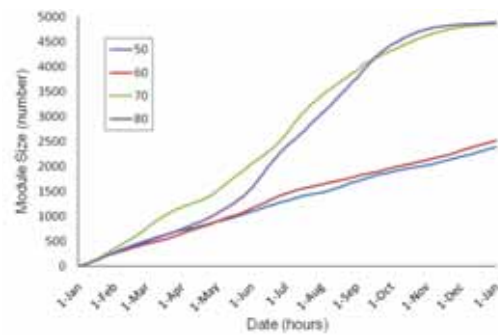


Fig. 8. Difference of OSS growth according to the change of inner motivation

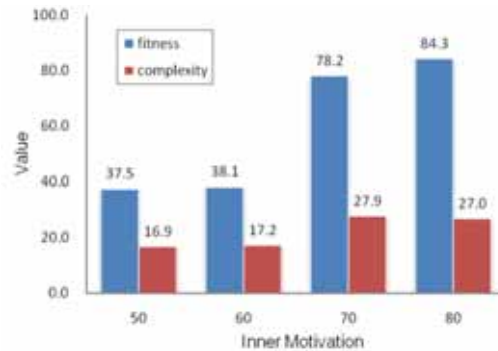


Fig. 9. Average Fitness and Complexity of OSS according to the change of inner motivation

The study results indicated that the file size increased linearly in the case of low inner motivation, but the file size increased rapidly after a certain time in the case of high inner motivation. We determined that this is caused by a difference in all of the developers' contribution time. Some people in the developer group contributed to the OSS and then seceded from it when they were bored, because their inner motivation was driven by personal hobbies and preferences. As shown in Figs. 8 and 9, in the case of low inner motivation the OSS grows linearly and the fitness is low because the developer group does not take an interest in it and their contribution time is short. However, in the case

of high inner motivation, the growth rate is higher as more developers conduct voluntary participation and recommendations, because the developer group takes an interest in the OSS and their contribution time is long.

Next, the parameter values for outer motivation changed from 1 to 7 by intervals of 2 and 10 simulation runs were executed for each case. Fig. 10 shows the difference in growth according to the change in the parameter values. Fig. 11 shows the difference in the average fitness and complexity of the OSS. We determined that this situation is caused by a difference in the general developers' contribution time due to the given parameters. The results of Figs. 10 and 11 indicate that the growth rate is higher and the code is more complete when the contribution time is longer due to high outer motivation.

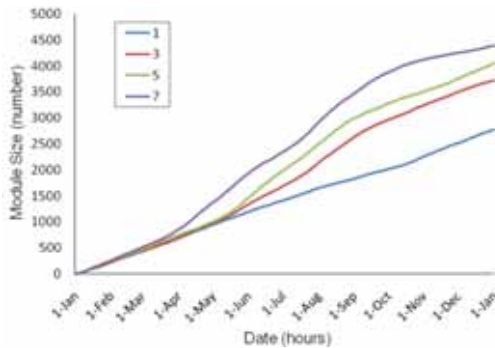


Fig. 10. Difference of OSS growth according to the change of outer motivation

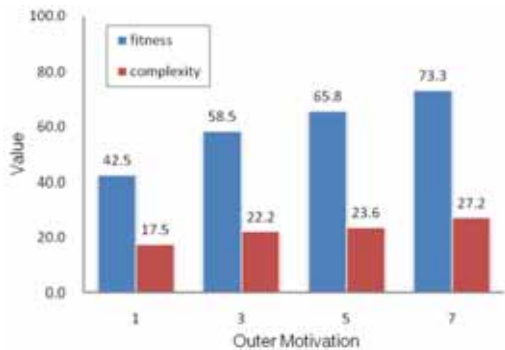


Fig. 11. Average Fitness and Complexity of OSS according to the change of outer motivation

### 5.3. Role of Core Developer and OSS Growth/Fitness

We studied how the role of core developers affects the growth and fitness of OSS. The parameter values for their active area changed from 5 to 13 by intervals of 2 and 10 simulation runs were executed for each case. Fig. 12 shows the difference in growth according to the change in parameter values and Fig. 13 shows the difference in average fitness and complexity according to this change.

These results indicate that the OSS growth rate and fitness are higher when the active area is larger, in other words, they effectively represent the mission and direction and reflect the opinions of the developer group.

We determined that this is caused by a difference in the general developers' contribution time due to the given parameters. When their active area is small, the outer motivation of the general developer group who participates in the OSS decreases rapidly. Accordingly, when the contribution time of the general developer group is shorter, they frequently secede from the OSS and then the growth rate and fitness decrease. When their active area is large, the contribution time of the general developer group is longer and the growth rate and fitness increase.

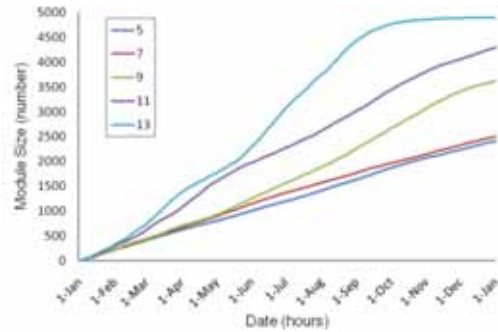


Fig. 12. Difference of OSS growth according to the change of active area

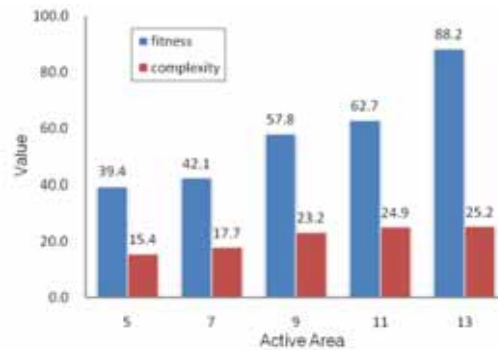


Fig. 13. Average Fitness and Complexity of OSS according to the change of active area

### 5.4. Role of User and OSS Growth

We studied how the role of the user affects the growth (expansion) of OSS. The parameter values for the users' interest in new features changed from 0% to 40% by 10% intervals and 10 simulation runs were executed for each case. Fig. 14 shows the difference in growth of the OSS according to the change in the parameter values and Fig. 15 shows the difference in its average fitness and complexity due to this change.

When the needs for new features are not frequent, Figs. 14 and 15 show that the expansion of the project is slow but the fitness and complexity of the OSS is high. On the contrary, when the needs for new features are frequent, they also show that the expansion of the project is fast, but the fitness and complexity of the OSS is low. As the need for new features increases due to frequent requirements from



the user group, the development area expands quickly, but the average fitness and complexity are controlled by the exchange of requests between the user group and the developer group.

We interpreted these results to indicate a difference in communication between the developer group and the user group. When the need for new features is not frequent, the developer group does not reflect the opinions of the user group and it only develops in the initial development area because of the lack of communication between the two groups. Therefore, its fitness may be higher but the various requirements of the user group are not satisfied, so the expansion rate of the OSS may gradually decrease. On the contrary, when the various requirements of the user group are frequent, the developer group develops the OSS to reflect the various requirements of the user group due to sufficient communication between the two groups and continuous evolution is possible.

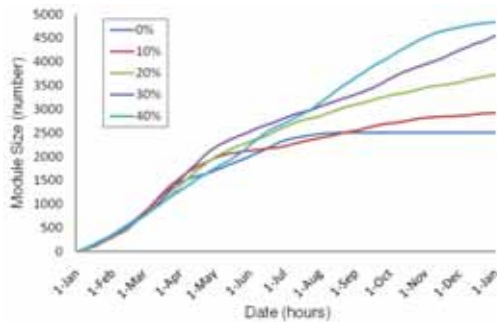


Fig. 14. Difference of OSS growth according to the change of request for new features

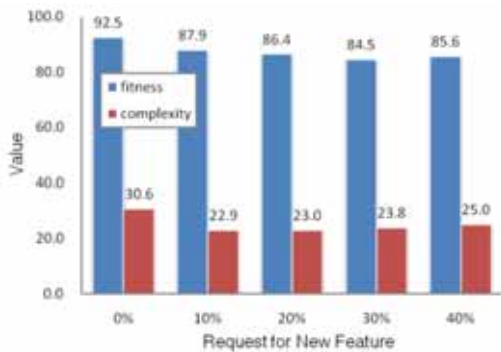


Fig. 15. Average Fitness and Complexity of OSS according to the change of request for new features

### 5.5. Refactoring Function and OSS Fitness/Complexity

We studied how refactoring affects the fitness and complexity of OSS. Ten simulation runs were executed for each case whether refactoring existed or not. Fig. 16 shows the difference in growth due to the changes in refactoring and Fig. 17 shows the difference in the average fitness and complexity of the OSS due to these changes. When refactoring task is implemented, the difference in the growth of the OSS is not large but the difference in its fitness and

complexity is relatively large. As a result of refactoring, the OSS complexity is decreased, which affects the fitness and total quality of the OSS. Without the refactoring task, the developer can find it difficult to understand the written codes since they are more complex.

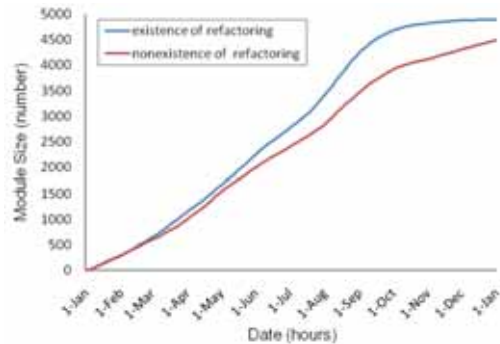


Fig. 16. Difference of OSS growth according to the existence of function of refactoring

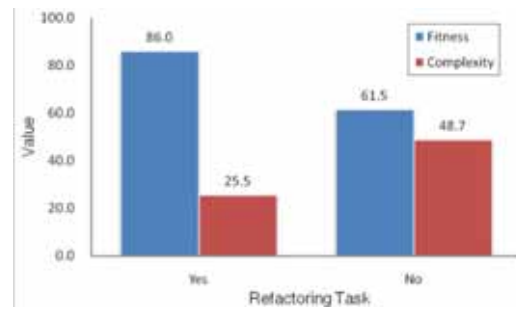


Fig. 17. Average Fitness and Complexity of OSS according to the existence of function of refactoring

## 6. Conclusions

According to Lehman's laws of software evolution, the software becomes gradually more difficult to add new modules or functions as a software development project gets larger and more complex [13]. This occurs because of the limited number of developers and the closed source code in the project. In the OSS, however, every developer can participate in the project due to the potential for an unlimited number of developers. Due to this, the code is written quickly and the OSS grows at a rapid rate.

The rapid growth of OSS occurs due to an ecosystem of developers and users and OSS itself. In other words, the growth of OSS does not occur due to a single factor, but by a combination of various factors among each agent. We can observe this phenomenon in our simulation model. If any factor of the OSS is weak or small, the rapid growth does not occur effectively.

Prior literature indicates that the evolution of OSS is focused on one or two factors. However, our article combines several factors found in prior literature and makes an agent-based model that uses many factors to explain the evolution process of OSS. From this model, we demonstrated that the factors affected by the OSS evolution

are the participation and duration of developers, participation of users and activity of core developers. Since we can trace what factors affect the evolution of OSS and how they affect its evolution, software development companies that want to participate in OSS development can improve their software development strategy by considering the results of our study. To achieve an effective outcome, the companies should understand the core factors of OSS. By considering these core factors, efficient investments can induce the effective development of OSS projects for the company. These investments will also lead to the successful evolution of OSS itself.

## References

- [1] Hars, A. and Ou, S. (2002), "Working for Free? Motivations for participating in open source projects", *Intern. J. Electronic Commerce*, 6(3).
- [2] Xu, J., Madey, G. (2004), "Exploration of the Open Source Software Community", NAACOSOS Conference.
- [3] Godfrey, M. and Tu, Q. (2001), "Growth, Evolution, and Structural Change in Open Source Software", *International Workshop on Principles of Software Evolution*.
- [4] Smith, N., Capiluppi, A., Fernandez-Ramil, J. (2006), "Agent-based Simulation of Open Source Evolution", *Softw. Process Improve. Pract.*, 11:423-434.
- [5] Long, J. (2006), "Understanding the Role of Core Developers in Open Source Software Development", *Journal of Information, Information Technology, and Organizations*, Vol 1.
- [6] Madey, G., Freeh, V., and Tynan, R. (2003), "Agent-based Modeling and Simulation of Collaborative Social Networks", *AMCIS2003*, Tampa, FL. August.
- [7] Hertel, G. Neidner, S., and Hermann, S. (2003), "Motivation of software developers in Open Source projects: and Internet-based survey of contributors to the Linux kernel", *Research Policy*, 32(7), 1159-1177.
- [8] Ghost, R. and Prakash, V. (2000), "The Orbiten Free Software Survey", *First Monday*, 5(7), July.
- [9] FLOSS(2002), "Free/Libre and Open Source Software: Survey and Study", *FLOSS Final Report*.
- [10] Hann, I-H., Roberts, J., Slaughter, S., and Fielding, R. (2002), "Economic Incentives for Participating in Open Source Software Projects", in *Proc. Twenty-Third Intern. Conf. Information Systems*, 365-372, December.
- [11] Mockus, A., Fielding, R., & Herbsleb, J.D. (2002), "Two Case Studies of Open Source Software Development: Apache and Mozilla", *ACM Transactions on Software Engineering and Methodology*, 11(3), 309-346.
- [12] Xu, J., Gao, Y., Christley, S., Madey, G. (2005), "A Topological Analysis of the Open Source Software Development Community", *Proceedings of the 38th Hawaii International Conference on System Sciences*.
- [13] M. M. Lehman, J. F. Ramil, P. D. Wemick, D. E. Perry, and W. M. Turski. (1997), "Metrics and laws of software evolution – the nineties view", *4th International Software Metrics Symposium*
- [14] AnyLogic, <http://www.xjtek.com/anylogic>

# Towards Merging Goal Models of Networked Software

Zaiwen Feng, Keqing He, Rong Peng, Jian Wang, Yutao Ma  
State Key Lab of Software Engineering, Wuhan University  
420072, Wuhan City, Hubei Province, China  
fengzaiwen@sina.com

## Abstract

*The ultimate goal of networked software is to realize mass customization. i.e., to satisfy the individualized requirements at a low cost and in a short time. Domain knowledge created by domain modeling provides essential reuse basis for mass customization. However, to meet individualized requirements, it is necessary to customize requirements based on common domain knowledge. Under this background, a number of individualized goal models are brought out. These models are difficult to reuse for user's individualized request and manage. In order to prompt knowledge reuse, merging individualized goal models is necessary. In this paper, we present an approach to merge goal models with high semantic similarity. We define three basic merging patterns to merge atomic goal models with involvement of human, and describe the respective algorithm. Based on basic merging patterns, a systematic algorithm is presented to solve more general merging.*

**Keywords:** pattern, goal, merging, knowledge

## 1. Introduction

Recently, service-oriented software development has been deemed as a new programming paradigm in software engineering to prompt the next revolution in software development [4]. Service-Oriented software is composed of loose-coupling component distributed on Internet, especially (semantic) web services. Thus, organizations are able to create and deploy new software applications agilely to satisfy rapidly changing users' requirements.

Based on the background, we propose Networked Software (NS) that is a complex system of which topology structure and activity can be evolutionary dynamically [5]. We proposed systematic development methodology and formalization frame for NS [6, 12]. In summary, the most important feature of NS development is to meet the common requirements of

user's via mass customization, and to meet individualized, diversiform requirements via fast-response, changing-with-demand services.

Developing NS consists of two basic phase. First is modeling selection according to user's requirements. It includes: text requirements are required on-line, and then requirements goal is elicited from text requirements. The system will query in the domain knowledge base (DKB) and give the solution (always represented as goal requirements model) as feedback to user's requirements. Nevertheless, solution that DKB provides may be always too generalized to fulfill individualized requirements. So the second phase is individualizing. User adds, deletes or modifies the initial solution based on common base knowledge to meet individualized goal. The result is: there will always be more than one varied goal models around a base goal model. As time goes by, these varied goal models become more and more and they are difficult to reuse and manage. So it is necessary to integrate all varied goal models to uniformed one in order to meet individualized requirements of user group.

In this paper we propose a systematic approach to merge semantic similar goal models based on RGPS. We generalized three refinement patterns for AND/OR-Refinements via observation. These refinement patterns are constraint condition for judging whether models can be merged. Then we present three types of merging patterns and corresponding algorithm. Lastly we propose a general algorithm for complex merging by means of invoking basic merging pattern. Our approach requires humans in some phase to ensure correctness of merging.

The paper is structured as follows: Section 2 describes definition of goal model in RGPS briefly and mapping method. Section 3 provides three refinement patterns of goal model which is the theory base of our paper. Section 4 depicts the pattern-based merging approach. Section 5 gives a case study for the approach. Section 6 discusses related works currently. At last, Section 7 concludes the paper.

## 2. Goal Model of NS

This Section we will give the definition about goal of NS and semantic similarity of goals.

### 2.1. Definition of Goal

A *goal* is an objective the system under consideration should achieve. Goals may be formulated at different levels of abstraction, ranging from high-level, strategic concerns (such as “*provide ubiquitous cash service*” for an ATM network system) to low-level, technical concerns “*card kept after 3 wrong password entries*” for an ATM system [9].

The goal layer is important layer of RGPS frame. In RGPS frame, goals include functional goals and non-functional goals. Functional goals describe function that a system must achieve. A functional goal consists of three parts in RFGS [12, 6]. That is, a verb that indicates the *operation*, a noun that indicates the *object* dealt with by the *operation*, and the *manner*, a prefix or a suffix that indicates how operation affects the object. Such as the functional goal “*Sort order by arrival of time*”, we can extract from it that *operation* is “*Sort*”, the *object* is “*order*”, and the *manner* is “*by arrival of time*”.

In goal-oriented methodology, goal is always elaborated from high-level to concrete operation and operational description of system-to-be. Goal refinement is a process that a high-level goal is decomposed into subgoals. Generally speaking, in goal-oriented methodology, goal refinement strategy is classified as AND-Refinement and OR-Refinement. AND-Refinement means that satisfying all subgoals in the refinement is sufficient for satisfying the parent goal. OR-Refinement means satisfying one of the refinements is sufficient for satisfying the parent goal [9]. In this paper, all goal models to be merged adopt AND-Refinement or OR-Refinement.

### 2.2. Semantic Similarity of Goal Definition

To merge goal requirements model, the first necessary step is to map concepts of goal in two models. The principle of mapping two goal concept is they are semantic similar. According to Section 2.1, a goal definition is divided to three parts:

DefinitionOfGoal = {*operation, object, manner*}

Necessary condition of semantic similarity of goal definitions is that each part of them are synonymy. For example, G1 “*Book train ticket*” and G2 “*Order train ticket*” have semantic similarity since *operation*

of G1 “*Book*” and *operation* of G2 “*Order*” are synonymy.

## 3. Refinement Patterns for Goal Model

In goal-oriented methodology, goal is always elaborated from high-level to concrete operation and operational description of system-to-be. [10] discusses goal refinement pattern from formal perspective with temporal logic. In this paper we present some refinement pattern from engineering perspective via generalization to a great deal of goal models from projects. See Table 1.

When domain expert edits knowledge of DKB, it will be his consideration scope that which refinements pattern is for each goal assertion to be decomposed. Extending to concept set of Section 2.1, we define definition set for each goal (not leaf goal) below.

DefinitionOfGoal = {*operation, object, manner, refinementpattern*}

## 4. Approach of Merging Goal Model

In the section we first give some definitions, then three basic merging patterns are presented, at last we depict a systematic algorithm for merging complex goal model.

### 4.1. Basic Definition

**Definition1 (Overlap Point).** *Overlap points* are a pair of goals which are respectively located in two goal models. Goal pairs must have semantic similarity.

**Definition2 (Merging Point).** *Merging point* is the joint for two goal models to be merged. Merging point is also overlap point. Generally roots of two models to be merged are *merging point*.

**Definition3 (Conflict Point).** When two goal models are merged, requirements semantic conflict may happen. We call a pair of goals which respectively lies in two goal models and conflict each other *conflict points*.

We use the techniques described in [11] to detect conflict between goals. General method consists of deriving boundary conditions by backward chaining, or the use of divergence patterns. Detection and resolution of goals requires involvement of human.

**Definition4 (Atomic Goal Model, AGM).** An AGM consists of goals and refinement relation. All non-leaf goals must have the same *refinement pattern*. Refinement relation must be all AND-Refinement simultaneously, or OR-Refinement simultaneously. Instances of AGMs are depicted in Figure 1.

Table.1. Refinement Pattern of Goal

<b>Refinement Patterns</b>	<b>Description</b>	<b>Example</b>
<i>Object Decomposition Pattern</i>	All subgoals are one part of the parent goal. If we decompose G to G1, G2,G3,...,Gn. Then we can say G1, or G2, or G3,..., or Gn is <i>one part of</i> Gn. Or G <i>includes</i> G1,G2,G3,...Gn. This decomposition pattern can be used in AND-Refinement.	Decomposing “Deal With Order” to “Add Order”, “Delete Order” and “Modify Order”. Decomposing “Obtain Information of City Facility” to “Obtain Information of hotel”, “Obtain Information of School”
<i>Business Process Decomposition Pattern</i>	Subgoals have temporal relation and could be regarded as a business process model. If we decompose G to G1,G2,G3,...,Gn, we can say G1, or G2, or G3,...,or Gn is one <i>subprocess</i> in achieving G. This decomposition pattern can be used in AND-Refinement.	Decomposing “Supply Customer” to “Get Order”, “Verify Order”, “Process Order”, “Package Order”, “Ship and Bill”.
<i>Means Decomposition Pattern</i>	All subgoals are means listed to address the parent goal. If we decompose G to G1,G2,G3,...,Gn, we can say G is achieved <i>by means of</i> G1,G2,G3,...,Gn (AND-Refinement). Or we can say G is achieved <i>by means of</i> G1, or G2, or G3,..., or Gn (OR-Refinement). This pattern can be used in AND or OR refinement.	Refine “Provide Feedback” to “Use Email” and “Use Web Form” with OR-Refinement. Refine “Ensure Secure Distance between Trains” to “Maintain Safe Speed”, “Maintain Safe Train Response to Command” and “Maintain no Sudden Stop of Preceding Train” with AND-Refinement.

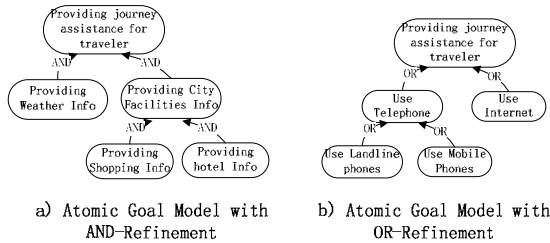


Fig.1. Instance of Atomic Goal Model

## 4.2. Basic Merging Pattern

**Definition5 (Basic Merging Pattern).** We define some *basic merging patterns*. When merging AGMs, these patterns help to make correct merging strategies. There are three *basic merging patterns*: AND-AND Pattern, OR-OR Pattern and AND-OR Pattern.

**Definition6 (Condition for merging).** Necessary condition for merging two AGMs is: (i) Semantic similarity for roots of AGMs; (ii) Refinement patterns of AGMs must be the same if both are AND-Refinement, or both are OR-Refinement. Refinement pattern can be different if one is AND-Refinement, and the other is OR-Refinement.

- AND-AND Pattern

When two AGMs to be merged are both AND-Refinement, simultaneously have the same refinement patterns, then AND-AND pattern will be used.

Supposing T and T' are goal models to be merged. G, G' are respectively root of T, T'. G, G' are *merging point*. The algorithm for merging model T and T' with pattern AND-AND is defined below:

Algorithm MergeWithAND-AND(T,T',G,G')

1. Depth-first traverse T' from G'.
2. For each goal  $G_j' \in T'$ . Supposing *overlap point* set of T and T' is set A ( $A = \{(G_i, G_j') | G_i \in T, G_j' \in T', SemanticSimilarity(G_i, G_j')\}$ ). If  $G_j'$  is not one of goals of goal pairs in A, then domain expert will determine where  $G_j'$  is inserted in T as merging. If  $G_j'$  is a leaf goal of T', then  $G_j'$  is added to T in a position that domain expert determines. If  $G_j'$  is not a leaf goal of T', then  $G_j'$  is added to T in a position that domain expert determines, and all subgoals of  $G_j'$  in T' will be still subgoals of  $G_j'$  in T.
3. Conflict detecting. When  $G_j'$  of T' is added to T, detect whether conflict happens between  $G_j'$  and other goals of T. So we get the set of *conflict point* B,

$$B = \{(Gi, Gj') \mid Gi \in T, Gj' \in T', GoalConflict(Gi, Gj')\}$$

. Resolve conflict if conflict happens. Extremely, merging fails if conflict cannot be resolved by all means.

4. If  $Gj'$  is one of goals of goal pairs in A and  $Gj'$  is a leaf goal, then  $Gi$  and  $Gj'$  are merged to the one in T (( $Gi$  and  $Gj'$  are *overlap point*)).
5. If  $Gj'$  is one of goals of goal pairs in A and  $Gj'$  is not a leaf goal, supposing  $Gi$  and  $Gj'$  are *overlap point*, P, P' are subtrees of T, T' as  $Gi, Gj'$  are roots. MergeWithAND-AND(P,P',Gi,Gj'). Recursion happens.

Thus, T becomes the new goal model after T' merges to T with AND-AND pattern.

### ● OR-OR Pattern

When two AGMs to be merged are both OR-Refinement, then OR-OR pattern will be used.

Supposing T and T' are goal models to be merged. G, G' are respectively root of T, T'. G, G' are *merging point*. The algorithm for merging model T and T' with pattern OR-OR is defined below:

Algorithm MergeWithOR-OR(T,T',G,G')

1. Depth-first traverse T' from G'.
2. For each goal  $Gj' \in T'$ . If  $Gj'$  is not one of goals of *overlap point*, then domain expert will determine where  $Gj'$  is inserted in T as merging. If  $Gj'$  is a leaf goal of T', then  $Gj'$  is added to T in a position that domain expert determines. If  $Gj'$  is not a leaf goal of T', then  $Gj'$  is added to T in a position that domain expert determines, and all subgoals of  $Gj'$  in T' will be still subgoals of  $Gj'$  in T.
3. If  $Gj'$  is one of goals of *overlap point* and  $Gj'$  is a leaf goal, then  $Gi$  and  $Gj'$  are merged to the one in T (( $Gi$  and  $Gj'$  are *overlap point*)).
4. If  $Gj'$  is one of goals of *overlap point* and  $Gj'$  is not a leaf goal, supposing  $Gi$  and  $Gj'$  are *overlap point*, P and P' are subtrees of T and T' as  $Gi$  and  $Gj'$  are roots. MergeWithOR-OR(P,P',Gi,Gj'). Recursion happens.

Thus, T becomes the new goal model after T' merges to T with OR-OR pattern.

### ● AND-OR Pattern

When we merge one AGM with AND-Refinement to the other with OR-Refinement, we will use AND-OR pattern.

**Case (a)** AGM T is AND-Refined with *Object decomposition pattern*, or *Business process decomposition pattern*, the other AGM T' is

OR-Refinement. G, G' are respectively root goals of T, T', and G, G' are *merging point*. The algorithm for case (a) is depicted below (See Figure 2).

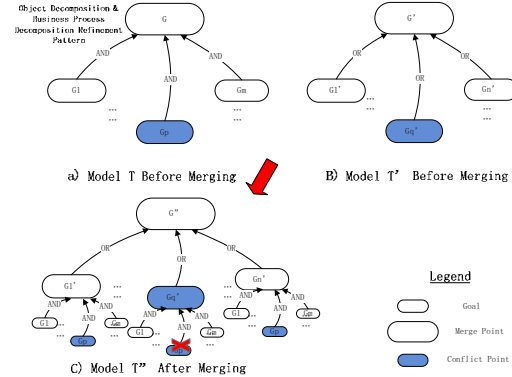


Fig.2. AND-OR Pattern (Case (a))  
Algorithm MergeWithAND-OR\_a(T,T',G,G')

1. Supposing G'' is the new goal after merging G and G'. Add all subgoals of T' ( $G1', G2', \dots, Gn'$ ) to G1'' with OR-Refinement. Then add all subgoals of T ( $G1, G2, \dots, Gm$ ) to  $G1', G2', \dots, Gn'$ . New goal model T'' (See Figure 2 (c)) is generated.
2. Conflict Detecting. When we add  $G1, G2, \dots, Gn$  to  $Gq'$ , conflict should be detected between  $G1$  and  $Gq'$ ,  $G2$  and  $Gq'$ , ...,  $Gn$  and  $Gq'$ . Supposing conflict is found between  $Gp$  and  $Gq'$  (See Figure 2 (c)). We first resolve the conflict between  $Gp$  and  $Gq'$ . Extremely,  $Gp$  will be deleted if conflict cannot be resolved by any means. Iterate the same operation from  $G1'$  to  $Gn'$ .

Algorithm ends. Thus we obtain new goal model T'' after merging T and T'.

**Case (b)** One AGM T is AND-Refinement with *Means Decomposition pattern*, the other T' is OR-Refinement. G, G' are respectively root goals of T, T', and G, G' are *merging point*. The algorithm for case (b) (See Figure 3) is depicted below.

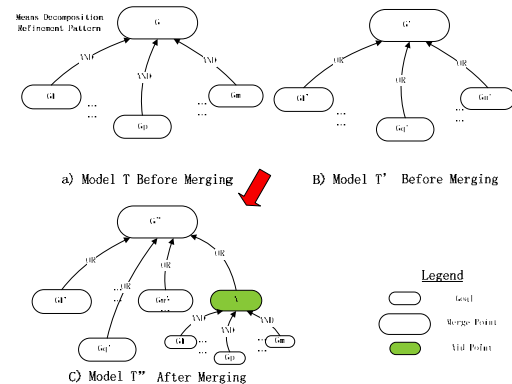


Fig.3. AND-OR Pattern (Case (b))  
Algorithm MergeWithAND-OR\_b(T,T',G,G')

1. Supposing  $G''$  is the new goal after merging  $G$  and  $G'$ . Add goal  $A$  to  $G''$  with OR-Refinement. The goal  $A$  is aid point.
  2. Add all subgoals of  $G$  ( $G_1 \dots G_m$ , see Figure 3 (a)) to aid point  $A$  with AND-Refinement.
- Algorithm ends. Thus we obtain the new goal model  $T''$  after merging  $T$  and  $T'$ .

### 4.3. Complex Merging

**Definition7 (Complex Merging).** If each model to be merged has more than one AGMs, we name the process of merging as *complex merging*.

Supposing  $G, G'$  is respectively the root of  $T, T'$ .  $T, T'$  are goal models to be merged.  $G, G'$  is *merging point*. The following is the algorithm for complex merging.

Algorithm MergingComplex( $T, T', G, G'$ )

1. Searching for all *overlap points* ( $G_i, G_j'$ ) of goal model  $T$  and  $T'$ . Supposing all subgoals of goal model  $T$  is in set  $A = \{goal_i | goal_i \in T \wedge goal_i \notin LeafNode(T)\}$ , and all subgoals of goal model  $T'$  is in set  $B = \{goal_i | goal_i \in T' \wedge goal_i \notin LeafNode(T')\}$ . Traverse  $A, B$  respectively and find goal pairs that satisfy  $G_i \in T \wedge G_j \in T' \wedge Semanticsimilarity(G_i, G_j')$ .
2. Traverse  $T, T'$  to search for AGMs of  $T, T'$ . We flag goals which belong to *overlap points* and are the roots of AGMs.
3. Supposing  $G''$  is the new goal after merging  $G$  and  $G'$ . Based on *basic merging pattern*, merging two AGMs which  $G, G'$  is respectively the root of. Supposing the output is goal model  $T''$  which  $G''$  is the root of.
4. Address the next flagged goal  $L$  of  $T$ . Supposing  $L, L'$  are *overlap points*. Merging the goal pairs ( $L, L'$ ) in  $T''$  based on *basic merging pattern*. Iterate this step until all flagged goals of  $T$  have been addressed.

Algorithm ends. The output  $T''$  is the merged goal model.

## 5. A Case Study

In this section, we illustrate *complex merging* with a case study in urban transportation domain.

Urban transportation query information system help travelers to arrange routine in city. E.g. [13]. See Figure 4.  $T, T'$  are both individualized goal models, and our task is to merge  $T$  and  $T'$ . Roots of goal models that is “Arrange Bus Travel Routine” are *merging point*.

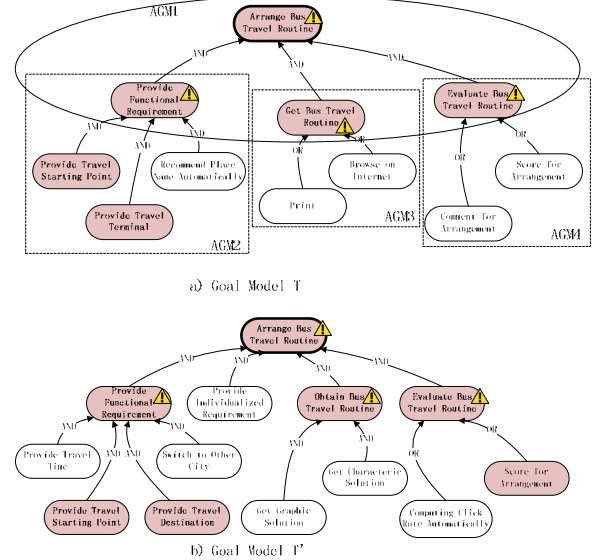


Fig.4. Case Study of *Complex Merging* (Before merging)

According to algorithm MergingComplex( $T, T', G, G'$ ), First we will search for *overlap points* of  $T, T'$ . See Figure 4, *overlap points* are signed with shallow red, and *merging point* is signed with shallow red and bold black border. Then traverse  $T, T'$ , we found AGMs of  $T, T'$  (Signed with rectangle or eclipse dashed border, see Figure 4(a), show omitted in Figure 4(b)). We flag goals in  $T, T'$  which are both roots of AGMs and belong to *overlap points* (Signed with exclamation mark in yellow triangle). Table 2 depicts refinement patterns of AGMs of  $T, T'$ .

Data of Table 2 shows that AGM1, ..., AGM4 of  $T, T'$  can all be merged respectively. Thus we merge AGM1 which the goal “Arrange Bus Travel Routine” are root of  $T, T'$  are merged to the new goal model  $T''$  after that. *Merging point* of  $T, T'$  is the goal “Arrange Bus Travel Routine”.

We continue to merge AGM2, AGM3 and AGM4 in  $T''$  with AND-AND pattern, AND-OR pattern and OR-OR pattern respectively. In the end we get the new merged model  $T''$ . See Figure 5.

Table.2. Atomic Goal Model of T, T'

	<i>T</i>		<i>T'</i>		<i>Merged Yes or No– (If Yes, Merging Pattern)</i>
	<i>Root of AGM – AND/OR Refinement</i>	<i>Refinement Pattern</i>	<i>Root of AGM – AND/OR Refinement</i>	<i>Refinement Pattern</i>	
AGM1	Arrange Bus Travel Routine-AND-Refinement	Business Process Decomposition Pattern	Arrange Bus Travel Routine-AND-Refinement	Business Process Decomposition Pattern	Yes (AND-AND Merging Pattern)
AGM2	Provide Functional Requirement-AND-Refinement	Object Decomposition Pattern	Provide Functional Requirement-AND-Refinement	Object Decomposition Pattern	Yes (AND-AND Merging Pattern)
AGM3	Get Bus Travel Routine-OR-Refinement	Means Decomposition Pattern	Obtain Bus Travel Routine-AND-Refinement	Object Decomposition Pattern	Yes (AND-OR Merging Pattern)
AGM4	Evaluate Bus Travel Routine-OR-Refinement	Means Decomposition Pattern	Evaluate Bus Travel Routine-OR-Refinement	Means Decomposition Pattern	Yes (OR-OR Merging Pattern)

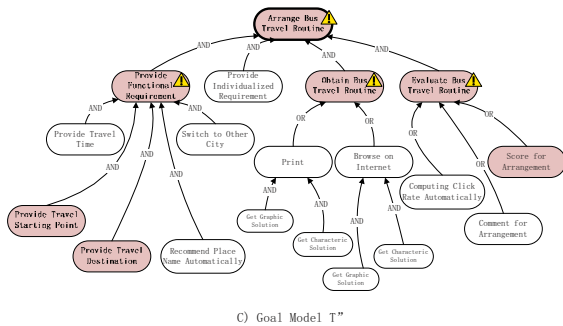


Fig.5. Case Study of *Complex Merging* (After merging)

## 6. Related Works

There are many works on mapping and merging of ontologies at present. It seems that there always are more than one ontologies in the same domain, which will cumber effective semantic queries for knowledge. So it is necessary to coordinate ontologies including mapping, alignment, and merging. HCONE-merge [7] can automatically align and then merge ontologies, HCONE-merge makes use of the intended informal meaning of concepts by mapping them to WordNet senses using the Latent Semantic Indexing method. ONION [14] present an Ontology-Composition Algebra that consists of a set of basic operators that can be used to manipulate ontologies. FCA-Merge [15] apply techniques from natural language processing and formal concept analysis to derive a lattice of concepts which is explored and transformed to the merged

ontology by the ontology engineering. These approaches are efficient to merge knowledge as ontologies but does not refer to merging of requirements model.

[8] proposes the Constraints-based Modular Petri Nets (CMPN) approach as an effective way to formalize the informal aspects of use cases. Further, it aims to integrate use cases from different viewpoints and analysis completeness and consistency.

Some works focus on merging of process model such as [16, 17]. [16] groups merges in several categories and describes the corresponding algorithm for performing these operations. Based on first order logic implemented by a set of Prolog rules, [17] proposed an approach for merging overlapping orchestration by defining a formal model named OMSM and guiding the developers with transformation rules to create new orchestration.

## 7. Conclusions

In this paper we propose a systematic approach aiming to merge individualized goal model. Based on observation to a number of goal models depicted in Tropos, Kaos, we group AND-Refinement & OR-Refinement relation in three refinement patterns: *Object decomposition pattern*, *Business process decomposition pattern* and *Manner decomposition pattern*. Then we present four types of *basic merging patterns*, corresponding algorithm are presented too. *AGMs* can be merged with one of *basic merging patterns* within corresponding constraints. Our approach requires domain expert such as: Validate the



inserted position in some merging patterns, detect and resolve conflict. In the end we propose a general algorithm for complex merging by means of invoking basic merging patterns.

Using this approach, domain experts can integrate a number of semantic similar individualized goal models to one uniform model which is applicable to a user group. Reuse degree of domain knowledge is enhanced. In this way, user can directly query the goal model that can cover his individualized requirements which is impossible to realize before merging.

## ACKNOWLEDGMENT

This research project was supported by the National Basic Research Program of China (973) under Grant 2007CB310801, the National High Technology Research and Development Program of China (863) under Grant No.2006AA04Z156, the National Natural Science Foundation of China under Grant No.60873083, 60703018, 60803025 and 60703009.

## References

- [1] P. Bresciani, A. Perini, P. Giorgini, "Tropos: An Agent-Oriented Software Development Methodology", *Journal, Autonomous Agents and Multi-Agent Systems*, Volume 8, Number 3, pp. 203-236, 2004.
- [2] J. Castro, M. Kolp, J. Mylopoulos, "Towards Requirements-Driven Information System Engineering: The Tropos Project", *Journal, Information System*, Vol 27, Issue 6, pp. 365-389, 2002.
- [3] B. Henderson-Sellers, P. Giorgini, P. Bresciani, "Enhancing Agent OPEN with concepts used in the Tropos methodology", Available at: <http://dit.unitn.it/~pgiorgio/papers/esaw03.pdf>
- [4] N. Gold, A. Mohan, C. Knight, et al "Understanding Service-Oriented Software". *Journal, IEEE Software*, 21(2): pp. 71-77, 2004.
- [5] K. Q. He, P. Liang, R. Peng, et al, "Requirement emergence computation of networked software", *Frontier of Computer Science in China*, 1(3): pp. 322-328, 2007.
- [6] K. Q. He, R. Peng, et al, "Networked Software", Chinese Science Press, ISBN: 978-7-03-023160-4, Beijing, China, 2008.
- [7] Konstantinos Kotis, George A. Vouros, Konstantinos Stergiou. "Towards automatic merging of domain ontologies: The HCONE-merge approach", *Journal, Web Semantics Science, Services and Agents on WWW*. 4(2006), pp.60-79.
- [8] Woo Jin Lee, Yong Rae Kwon. "Integrating and Analysis of Use Cases Using Modular Petri Nets in Requirements Engineering", *Journal, IEEE Transaction on Software Engineering*, VOL. 24, NO.12, Dec 1998.
- [9] Axel van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", In 5th IEEE International Symposium on Requirements Engineering, Toronto, August 2001, pp. 249-263.
- [10] Robert Darimont, Axel van Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration", In: Proceedings 4th ACM Symposium on the Foundations of Software Engineering (FSE4), San Francisco, Oct. 1996, pp. 179-190.
- [11] Axel van Lamsweerde, "Managing Conflicts in Goal-Driven Requirements Engineering". *Journal, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, VOL. 24, NO. 11, NOVEMBER 1998.
- [12] Jian Wang, Keqing He, Ping Gong, et al "RGPS: A Unified Requirements Meta-Modeling Frame for Networked Software", In Proc. of Third International Workshop on Advances and Applications of Problem Frames (IWAAPF'08), Leipzig, Germany, May 2008.
- [13] Chendu Internet Services for People's Travel. Available at: <http://www.cdgzcx.com/index.jsp>
- [14] Prasenjit Mitra, "An Algebraic Framework for the Interoperation of Ontologies", A dissertation for the degree of doctor in Stanford University, August, 2004.
- [15] Gerd Stumme, Alexander Maedche. "FCA-Merge: Bottom-Up Merging of Ontologies". In 7th Intl. Conf. on Artificial Intelligence (IJCAI '01), Seattle, WA, 2001. pp. 225-230.
- [16] Shuang Sun, et al Merging workflows: A new perspective on connecting business processes. *Journal of Decision Support Systems*. 42(2006) pp.844-858.
- [17] Clémentine Nemo-Cailliau1, Tristan Glatard1, Mireille Blay-Fornarino, et al. "Merging overlapping orchestrations: an application to the Bronze Standard medical application". In IEEE International Conference on Services Computing (SCC 2007), Salt Lake City, USA, July 2007. pp. 364-371.

# Comparison of Some Single-Agent and Multi-Agent Information Filtering Systems on a Benchmark Text Data Set

Snehasis Mukhopadhyay\*, Shengquan Peng\*, Rajeev Raje\*, Mathew Palakal\*, and Javed Mostafa<sup>†</sup>

**Abstract**—Information filtering is a technique to identify, in large collections, information that are relevant according to some criteria (e.g., a user’s personal interests, or, a research project objective). There have been many information filtering systems developed by many researchers using a variety of techniques. The authors of this paper have also developed three information filtering systems: SIFTER (Smart Information Filtering Technology for Electronic Resources), D-SIFTER (Distributed SIFTER), and SIFTER-II. While SIFTER involves a single monolithic agent, D-SIFTER and SIFTER-II are developed using different multi-agent technologies. The objective of this paper is report the experimental results obtained with respect to the filtering performance and processing time for SIFTER, D-SIFTER, SIFTER-II, as well as several other filtering systems developed by other research groups on a benchmark data set, i.e., the OHSUMED collection available as part of the TREC-9 Ninth Text Retrieval Conference in 2005. The primary conclusion of these experimental results is that the multi-agent systems achieve comparable high filtering performance of single-agent monolithic systems, but with a drastically reduced processing time.

## I. INTRODUCTION

Advances in computer networking have made it possible for an increasingly wider spectrum of the population to access and manipulate digital information. Paralleling this development has been the exponential growth in the volume of information available through world-wide computer networks. As the size of the interconnected world will grow, so will be the burden on users in selecting relevant information from the flood of solicited or unsolicited information. How to identify relevant information from large collections and prevent an “information overload” on user is the focus of many research and commercial efforts. Information filtering (IF) is a technique used to classify, sort, and present information according to a particular user’s interest. IF systems are commonly personalized to support long-term information needs of a particular user or a group of users with similar needs. They accomplish the goal of personalization by directly or indirectly acquiring information from the user. In IF systems, these long-term information needs are represented as interest profiles (Lewis, 1995), which are subsequently used for matching or ranking purposes.

SIFTER (Mostafa *et al*, 1997) is a single agent centralized information filtering system, aiming to tackle the information overload problem. A single-agent information filter faces serious problems while dealing with relatively

large-scale applications. Creating a single monolithic large filter serving the users over a large information domain leads to unacceptable processing time, poor fault tolerance, and poor adaptability. A collaborative society of agents involved in information filtering may overcome many of these limitations, while resulting in a filtering performance very similar to that of a single large monolithic agent. D-SIFTER (Distributed SIFTER) (Mukhopadhyay *et al*, 2005) provides this enhancement to SIFTER. D-SIFTER is a homogeneous system, in which all the agents are same except for their knowledge base. D-SIFTER suffers from the problems associated with scalability, and flexibility. Despite the limitations, D-SIFTER emphasizes the importance of a distributed filtering paradigm. These limitations of D-SIFTER are eliminated in SIFTER-II (Distributed Multiple Agent Information System) (Mukhopadhyay *et al*, 2005). The agents in SIFTER-II are heterogeneous, i.e., they have different functionality. In particular, different agents perform the various information tasks, e.g., document collection, document classification, and user interaction (interest profile maintenance and updating). The system is flexible and open, it allows agents to join and leave freely. All these differences make SIFTER-II a new and improved system. It retains the advantages of D-SIFTER, and adds more powerful features.

A critical question that arise in the design of large-scale information filtering systems is whether or not it is better to use distributed multi-agent information filtering. If so, quantitative results indicating such benefits in real-world situations are desirable. This is the main objective and contribution of this paper. Large-scale experimental studies involving computer science data-set and the well-known TREC data-set are presented to illustrate the advantage of distributed filtering as well as to compare the different approaches.

## II. A BRIEF OVERVIEW OF THE RELEVANT INFORMATION FILTERING SYSTEMS

In this section, we provide a brief description of the seven information filtering systems whose performances are compared on the TREC-9 benchmark data set. Out of these seven systems, three (SIFTER, D-SIFTER, and SIFTER-II) have been developed by the authors of this paper and the others have been reported in the on-line proceedings of the TREC-9 conference.

### A. Single-agent Filtering System (SIFTER)

SIFTER classifies and presents the incoming documents to the user in an ordered fashion based on the user’s interests. The agent maintains a knowledge base, called the thesaurus,

\* With the Department of Computer and Information Science, Indiana University Purdue University Indianapolis, 723 W. Michigan St. SL 280, Indianapolis, IN 46202 smukhopa@cs.iupui.edu

<sup>†</sup> With the School of Information and Library Sciences, University of North Carolina, Chapel Hill, NC 27599 jm@unc.edu

which consists of key words and phrases culled from an authoritative source in a specific domain of interest. The agent classifies incoming document based on the thesaurus into different categories, keeps on learning the user's interests for different categories and maintains a dynamic user profile. SIFTER is a single agent isolated system, consists of four components: a document representation module, a document classification module, a user profile learning module and a user interface module. The representation module is responsible for converting a document into a numeric structure that can be manipulated by the classification module. The well-known vector-space model is used for the representation (Salton, 1989). The classification module consists of two important stages: an unsupervised cluster learning stage and a vector classification stage. During the first stage, clusters are generated from an initial set of sample documents vectors and each is represented by its centroid. A simple heuristic unsupervised clustering algorithm, called Maximin-Distance Algorithm (Tou, 1974), is used to determine the centroids over the document vector space. During the second stage, the incoming documents are classified into the corresponding cluster according to the similarity between the document and the centroid. The measure used for computing the similarity between two document vectors is the cosine similarity measure (Salton, 1989). The user profile is used to determine the user preference for the different classes of information so as to prioritize the presentation of incoming document. A reinforcement algorithm is employed for user interest profiling. The user interface module provides user with a window, from which the user can interact with the system. More details can be found in (Mostafa *et al*, 1997).

#### B. Distributed Information Filtering System (D-SIFTER)

D-SIFTER is aimed at improving the information filtering performance by providing a collaborative environment so that the agents can help each other to complement the thesaurus deficiency. In particular, all the agents are identical, employing the same algorithms for document representation, classification and learning of user profiles, except for the thesaurus. This may be caused by difference in the domains of information that the agents are designed for, personalization of each agent's thesaurus to different users, or independent automatic term discovery process in the agents. The communication among different agents takes place through a shared server. The basic idea behind D-SIFTER is: when an agent fails to classify a document, it will put this document into a waiting queue on the server. If another agent classifies that document, the result will be placed in a result queue on the server. The original agent will periodically check the result queue and it will bring the result back (Raje *et al*, 1997).

D-SIFTER emphasizes the importance of a distributed filtering paradigm, and improves the system performance. However, D-SIFTER is a homogeneous system, in which all the agents are same except for their knowledge base. The agents can not communicate with each other directly, as they go through a central server, which results in the problems

associated with scalability, performance, and flexibility.

#### C. Distributed Multi-agent Information Filtering (SIFTER-II)

The agents in SIFTER-II (Mukhopadhyay *et al*, 2005) are heterogeneous, i.e., they have different functionalities. In particular, different agents perform the various information tasks, e.g., document collection, document classification, and user interaction (interest profile maintenance and updating). The communication method is flexible, i.e., the agent can choose a suitable method according to its intention. For example, when an agent wants to advertise a task to the agent community, it will broadcast the message; when an agent decides to coordinate with a specific agent, it will communicate with this agent directly. The document classification process can be carried out in parallel. The user can enhance the agent's knowledge base and share with other user agents as needed. The system is flexible and open, it allows agents to join and leave freely. All these differences make SIFTER-II a new and improved system. It retains the advantages of D-SIFTER, and adds more powerful features.

SIFTER-II has many different types of agents and distributed object services. These agents can be classified into five categories according to their functionalities: administrator agent, domain agent, wrapper agent, user agent and classifier agent. In addition, there is a centroid generator service and a sifter server.

The administrator agent provides the directory service to the SIFTER-II system. This agent provides all the information of the non-agent services, such as the training service. Each domain agent concentrates on a single domain, such as computer science or biomedical science. Each wrapper agent is responsible for retrieving documents from a specific source and transforming the information to a standard form. If there are new documents, the wrapper agent will notify the domain agents about these documents. The domain agents will broadcast the new documents to user agents. The user agent is the proxy of the user. Each user has a corresponding user agent. The user agent keeps a user profile and updates it by using user's feedback. The user agent is also responsible for coordinating with the domain agent to get new documents and with classification agent to classify the documents. The user can expand the default knowledge base or create a new one, and share their own knowledge with other user agents. The classification agent is in charge of classifying the documents. It has a representation and classification module, but does not have any knowledge base associated with it. This architecture lets the classification of multiple documents to work in parallel, not over-loading any agents.

#### D. The Fudan Filtering System

Fudan is a single-agent filtering system whose methodology and performance has been described in the TREC-9 papers (Wu *et al*, 2001). Very briefly, the system constructs an initial profile vector consisting of a weighted sum of a topic vector and feature vectors. The topic vector represents a set of important terms or words representing a topic of interest.

A feature vector is an additional set of words extracted from relevant documents so as to maximize the mutual information with the topic terms. When a new document is encountered, its similarity is computed with the profile vector using the well-known Cosine similarity measure. The new document is considered relevant if the similarity exceeds a user-defined threshold. The details of the methods can be found in (Wu *et al*, 2001).

#### E. The Microsoft Filtering System

The Microsoft filtering system is another one for which results were submitted for TREC-9 Conference (Robertson and Walker, 2001). Unlike SIFTER, D-SIFTER, and SIFTER-II, it does not classify documents, but merely computes a relevance value of documents, based on the occurrence of terms in a profile. The latter is adapted based on identifying “important” terms using a suitably defined measure of importance. The threshold used for determining which documents are relevant, is also adapted on-line. The details of the system can be found in (Robertson and Walker, 2001). However, the authors clearly state that the computational load was heavy, requiring a week’s time on a single machine for the particular task in TREC-9.

#### F. The CMU-Y Filtering System

(Zhang and Callan, 2001) discusses the details of the CMU-Y filtering system referred to in the experimental studies reported in this paper. Briefly, this filtering system has three major modules: YParser, YClipset and YLearner. YParser processes the input data stream, YClipset filters the input data based on a profile, and YLearner updates the profile based on relevance feedback. The initial profile is the set of terms used in the title and description fields of the topic of interest. The profiles are updated using the Rocchio algorithm (Rocchio, 1971). The dynamic profile is matched with a new document and an adaptive threshold is used to decide whether or not the document is relevant. The details of the various methods used in the system can be found in (Zhang and Callan, 2001).

#### G. The KAIST Filtering System

(Lee *et al*, 2001) describes the results of the experiments performed with the filtering system referred to as KAIST. In this system, once again the Rocchio algorithm (Rocchio, 1971) is used to update the profile and a support vector machine (SVM) algorithm is used as a pattern recognizer deciding whether a document is relevant or not. The results of the SVM classification are re-filtered using profile-document similarity, based on intra-class and inter-class thresholds.

It is clear that the SIFTER, D-SIFTER, and SIFTER-II systems are conceptually different from the others in the sense that they involve the intermediate step of document classification using unsupervised clustering, before profile learning using relevance feedback. Further, D-SIFTER and SIFTER-II are the only two systems referred to in this paper which use multi-agent technologies.

### III. EXPERIMENTS WITH TREC-9 INFORMATION FILTERING TRACK DATA

In order to find out whether or not it is better to use distributed multi-agent information filtering, large-scale experimental studies need to be conducted on a standard real-world data set to compare the performance of different approaches. This paper mainly describes the experiments conducted on SIFTER, D-SIFTER and SIFTER-II with TREC-9 Information Filtering Track data (OHSUMED document collection). The OHSUMED document collection is briefly described first. The experimental results with SIFTER, D-SIFTER, and SIFTER-II are presented and compared with other reported results (with Fudan, Microsoft, CMU-Y, and KAIST filtering systems) in the on-line TREC-9 conference proceedings.

#### A. OHSUMED document collection

The OHSUMED training collection is a set of 54,710 references from MEDLINE, the on-line medical information database, consisting of titles and/or abstracts from 270 medical journals published during 1987. The OHSUMED test collection is a set of 293,856 references from MEDLINE, published over a four year period (1988-1991). The available fields are title, abstract, MeSH indexing terms, author, source, and publication type. William Hersh (hersh@OHSU.EDU) and colleagues obtained the OHSUMED document collection for their information retrieval experiments (Hersh, 1994). Some abstracts are truncated at 250 words and some references have no abstracts at all (titles only).

#### B. Evaluation of System Performance

For the TREC experiments, filtering systems are expected to make a binary decision to accept or reject a document for each profile. Therefore, the retrieved set consists of an unranked list of document. Two measures were used in TREC-9 conference. One was essentially the linear utility measure, the other is precision-oriented measure (Robertson and Hull, 2001).

The linear utility measure has been described in previous TREC reports. The particular parameters being used are a credit of 2 for a relevant document retrieved and a debit of 1 for a non-relevant document retrieved:

$$Utility = 2 * R_+ - N_+$$

where  $R_+$  is the number of relevant documents and  $N_+$  is the number of non-relevant documents. When evaluation is based on utility, it is difficult to compare performance across topics. Simple averaging of the utility measure gives each retrieved document equal weight, which means that the average scores will be dominated by the topics with large retrieved sets. Furthermore, the utility scale is effectively unbounded below but bounded above; a single very poor query might completely swamp any number of good queries. On TREC-9 conference, another performance measure, T9U, is used as:

$$T9U = Max(2 * R_+ - N_+, MinU)$$

$MinU = -100$  for OHSU topics,  $-400$  for MeSH topics  
 $MnT9U$  is the mean value of the T9U measure over topics.

TABLE I  
COMPARISON OF FILTERING PERFORMANCE OF SINGLE-AGENT  
SIFTER, D-SIFTER, AND SIFTER-II WITH THE BEST REPORTED  
TREC9 RESULTS

SYSTEMS	MnT9P	MnT9U	Proc. time/doc(mSec)
Fudan	31.7	-1.1	NA
Microsoft	30.5	-5.3	NA
CMU-Y	26.1	-26.9	NA
KAIST	20	12.2	NA
SIFTER (theta 0.6)	30.6	-6.5	6165.8
D-SIFTER(3 agents)	29.9	-8.5	1500.7
D-SIFTER(6 agents)	28.8	-11.5	374.1
D-SIFTER(9 agents)	25.5	-23	162.7
D-SIFTER(12 agents)	22.9	-35	97.4
SIFTER-II(3 agents)	29.1	-10.5	1602.5
SIFTER-II(6 agents)	27.7	-14	523.2
SIFTER-II(9 agents)	24.4	-26.5	329.4
SIFTER-II(12 agents)	22	-38.5	192.1

(NA means 'not available')

The idea of precision-oriented measure is to set a target number of documents to be retrieved over the period of the simulation; the target is set to 50 documents for each topic. The measure, T9P defined below, is essentially precision, but with a penalty for not reaching the target:

$$T9P = M/Max(Target, N)$$

Target = 50 documents

M = Number of relevant retrieved documents

N = Number of retrieved documents

MnT9P is the mean value of the T9P measure over topics.

The comparison of the results obtained with the seven filtering systems is shown in Table 1. The results for SIFTER, D-SIFTER, and SIFTER-II were generated locally through extensive experimentation, while those for Fudan, Microsoft, CMU-Y, and KAIST were collected from the TREC-9 conference web site. It can be seen that the best performances with all of SIFTER, D-SIFTER, and SIFTER-II are comparable to the best filtering results reported in TREC-9. The thresholding parameter in the clustering algorithm (and hence, the number of centroids) in SIFTER was adjusted to realize the best filtering performance. Further, the number of agents in the distributed filters D-SIFTER and SIFTER-II were adjusted so as to realize as fast processing as possible, without sacrificing the filtering performance significantly (the main objective of multi-agent filtering). Since processing time results were not reported for TREC9 conference in (Robertson and Hull, 2001), we report only the average processing time per document realized with SIFTER, D-SIFTER, and SIFTER-II for comparable filtering performance. The results clearly show that much faster processing is possible with distributed multi-agent filtering systems. It is worth noting that, although precise processing time information was not provided for TREC-9 systems, the authors of the Microsoft system does report a week's continuous processing on a single machine for all the documents, which roughly corresponds to 11 secs or 11,000 msec of processing time per document.

### C. Analysis of the Results with the TREC Experiments

It can be concluded that, the advantages of distributed approaches over a centralized are lower processing time, even while maintain high filtering performance (as measured by precision and recall). With the measurements of TREC-9 conference, the filtering performance is comparable to the best reported results. This implies that these three information filtering systems all can give good filtering performance. Two approaches that incorporate distribution with respect to knowledge and functionality highlight measurable advantages of a distributed approach over a centralized approach. D-SIFTER and SIFTER-II are flexible and efficient information filtering system. They provide the necessary flexibility, adaptability and scalability, thereby, lending themselves to be implemented as a large interconnected system.

### REFERENCES

- [1] Fisher, G. and Stevens, C. (1991). Information access in complex, poorly structured information spaces. In *Proceedings of ACM Special Interest Group on Human Computer Interaction Annual Conference*, pages 63–70.
- [2] Hersh, W. R., Buck, C., Leone, T. J., and Hickam, D. H. (1994). Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual SIGIR Conference*, pages 192–201.
- [3] Lee, K., Oh, J., Huang, J., Kim, J., and Choi, K. (2001). TREC-9 Experiments at KAIST: QA, CLIR, and Batch Filtering. [http://trec.nist.gov/pubs/trec9/t9\\_proceedings.html](http://trec.nist.gov/pubs/trec9/t9_proceedings.html).
- [4] MESH. (2001). <http://www.nlm.nih.gov/mesh/>.
- [5] Mostafa, J., Mukhopadhyay, S., Lam, W., and Palakal, M. (1997). A multilevel approach to intelligent information filtering: Model, system, and evaluation. *ACM Transactions on Information Systems*, pages 368–399.
- [6] Mukhopadhyay, S., Peng, S., Raje, R., Mostafa, J., and Palakal, M. (2005) Distributed Multi-Agent Information Filtering: A Comparative Study. *Journal of the American Society for Information Science and Technology*, vol 56, no. 8, pp. 834–842.
- [7] Raje, R., Mukhopadhyay, S., Boyles, M., Patel, N., and Mostafa, J. (1997). On designing and implementing a collaborative system using java-rmi. In *Proceedings of the Fifth International Conference on Advanced Computing*, pp. 404–411.
- [8] Raje, R., Mukhopadhyay, S., Boyles, M., Papiez, A., Patel, N., Palakal, M., and Mostafa, J. (1997). A bidding mechanism for web-based agents involved in information classification. *WWW Journal, Special Issue on Distributed World Wide Web Processing: Applications and Techniques of Web Agents*, 1:155 – 165, 1998.
- [9] Rocchio, J. J. (1971). Relevance feedback in information retrieval in The SMART Retrieval System. *Experiments in Automatic Document Processing*, pages 313–323, Prentice Hall Inc.
- [10] Robertson, S. and Hull, D. A. (2001). The TREC-9 Filtering Track Final Report. [http://trec.nist.gov/pubs/trec9/t9\\_proceedings.html](http://trec.nist.gov/pubs/trec9/t9_proceedings.html).
- [11] Robertson, S. and Walker, S. (2001). Microsoft Cambridge at TREC-9: Filtering track. [http://trec.nist.gov/pubs/trec9/t9\\_proceedings.html](http://trec.nist.gov/pubs/trec9/t9_proceedings.html).
- [12] Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley.
- [13] TREC. (2001). <http://trec.nist.gov/data.html>.
- [14] Tou, J. T., and Gonzalez, R. C. (1974). *Pattern Recognition Principles*. Addison-Wesley.
- [15] Wu, L., Huang, S., Guo, Y., Liu, B., and Zhang, Y. (2001). FDU at TREC-9: CLIR, Filtering and QA tasks. [http://trec.nist.gov/pubs/trec9/t9\\_proceedings.html](http://trec.nist.gov/pubs/trec9/t9_proceedings.html).
- [16] Zhang, Y. and Callan, J. (2001). YFilter at TREC-9. [http://trec.nist.gov/pubs/trec9/t9\\_proceedings.html](http://trec.nist.gov/pubs/trec9/t9_proceedings.html).

# Towards Adaptable BDI Agent: a Formal Aspect-Oriented Modeling Approach

Lily Chang and Xudong He

School of Computing and Information Sciences, Florida International University

Miami, FL 33199, USA

Email: {lchan003, hex}@cis.fiu.edu

## Abstract

*In this paper, aspect-oriented concept is incorporated into predicate transition nets to model agents based on BDI structure, which describes the mental attitudes of autonomous agents. Our modeling approach not only explicitly models the BDI structure to bridge the gap between agent theory and agent design, but also modularizes the BDI agent model into various aspects to enhance the adaptability and reusability of agent models.*

## 1. Introduction

Multi-agent systems [21] have drawn a lot of attentions due to their modularity and adaptability in complex system design. On formal approach for agent-oriented modeling [9], there were many works based on Petri nets to model multi-agent systems. However, most of the works were focused on modeling agent mobility and controls at system level [13, 23]. There are essential concerns such as agent reasoning, behavior adaptation and interaction in multi-agent system modeling have not yet been fully addressed. In order to address the essential concerns of agent-oriented modeling, we adopt the well known Belief-Desire-Intention (BDI) model [19] as the fundamental structure of our Predicate Transition nets (PrT nets) [7] agent model. BDI model has been widely used in a number of rigorous logic models and applications [4, 12, 18, 20, 22]. Nevertheless, only a few of them provide concrete models. The *beliefs* refer to the knowledge that an agent has about the world; *desires* are the goals that an agent would like to achieve; and, *intentions* are the plans that can reach agent goals. Beliefs, desires and intentions describe the mental attitudes of an agent and are the key elements to enable rational actions, which address the property of agent autonomy [22]. In this paper, we use PrT nets as the modeling language and show that it is viable to explicitly model BDI structure. PrT nets are an excellent formal model for the study of critical aspects in concurrent and distributed systems. Especially, the non-determinism of PrT nets is an important feature in modeling agent autonomy and behavior adaptation. Since PrT nets are formal models that provide a sound basis for system analysis to unveil errors and missing requirements at the earlier stage of system development process, costly fixes at later stages can be avoided.

In addition to address the essential concerns of modeling an agent, we further exploit the modularity and adaptability of BDI model by introducing the concept of *aspect* from aspect-oriented programming (AOP) [10] into PrT nets. Although multi-agent system architecture has the advantage of modularity by decomposing a complex system into multiple agents that can be designed individually to solve particular problems, the modularity is addressed at system level since an agent is the unit of abstraction that encapsulates its functionalities and controls [21]. In AOP, aspects are non-functional properties that can be wrapped into modular units and used wherever necessary. As a result, the software system using AOP is more manageable and efficient [10]. Similarly, we consider the essential concerns regarding certain agent properties as the candidates of *aspects* in a BDI agent model. For example, autonomy is an essential property of an agent [9], that is, an agent is a decision maker instead of a passive object. During the execution of an agent task, agent may need to reason for the consequent in order to act coherently. Therefore, the behavior of reasoning can be modeled as a reusable aspect in addition to action model. Separation of reasoning concerns allows the extensibility of decision logic. The conceptual model of our aspect-oriented BDI agent is shown in Figure 1, where the crosscutting behaviors of an agent's action model are related to several essential agent concerns such as reasoning, behavior adaptation and interaction. Note that, in addition to BDI structure, our conceptual model includes an interaction aspect to address the modeling of an agent's interactions with external environment (e.g., resource acquisition). Although, modeling the interactions of multiple agents is not the focus of this paper, the interaction model is indispensable since sociality is an essential concern of agent-oriented modeling [9].

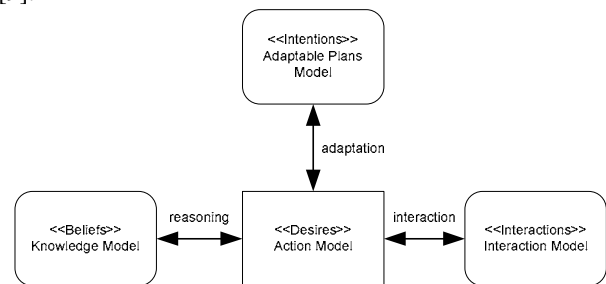


Figure 1. An aspect-oriented BDI agent model.

Other than essential aspects, there are different concerns with regard to different design objectives of agents. For instance, mobile agent design concerns about agent mobility, task agent design concerns about collaboration, interface agent design concerns about learning, etc. By separating different agent concerns into aspects from agent model, modularity and adaptability can be achieved at agent level. Consequently, the agent model is adaptable to incorporate different aspects for the analysis of different concerns and is more manageable to deal with extensibility.

The remainder of this paper is structured as follows. Section 2 presents the PrT nets modeling approach of aspect-oriented BDI agent model. Section 3 discusses the related works and the conclusion is drawn in Section 4.

## 2. Realizing Aspect-Oriented BDI Agent Model

In the following sections, we first introduce the PrT nets, and then present the modeling approach of BDI structure; lastly, we give the modeling approach of aspects, which includes specifying an aspect, aspect weaving, weaving patterns and weaving process.

### 2.1 Predicate Transition Nets

PrT nets are high level nets that are able to differentiate tokens by defining different token types and transition constraints. As a consequence, both data and controls can be addressed. More importantly, with operational semantics, PrT nets are amenable for model execution [5] and model checking [1]. A PrT net structure is composed of places, transitions and flow relations (arcs). Each place in a net can be defined to hold data tokens with a specified token type; and, each transition defines the enabling conditions in first-order logic formulas to select desired data tokens. Markings are states, which are token distributions in a net structure. Formally, a PrT net is a tuple  $(N, Spec, ins)$ , where  $N=(P, T, F)$  is a net structure.  $P$  and  $T$  are finite sets of places and transitions of  $N$ , where  $P \cap T = \emptyset, P \cup T \neq \emptyset$  and  $F \subseteq (P \times T) \cup (T \times P)$  is a set of arcs, which define the flow relations.  $Spec$  is an algebraic specification, which includes sorts, operators, and equations. Terms defined in  $Spec$  include tokens in  $P$ , labels on  $F$  and constraints associated with  $T$ . An inscription  $ins=(\varphi, L, R, M_0)$  maps net elements to their denotations in the algebraic specification  $Spec$ .  $\varphi$  is a mapping from  $P$  to the set of sorts;  $L$  is a sort-respecting mapping from  $F$  to the set of labels;  $R$  is a mapping from  $T$  to the set of constraints; and  $M_0$  is the initial marking – a mapping from  $P$  to the set of tokens. The formal definitions with regard to the dynamic semantics of a PrT net can be found in our previous work [3].

As an example for demonstration, a PrT net structure is shown in Figure 2, where the circles are places, the bars are transitions and the arrows are flow relations. By defining the net elements  $(P, T, F)$  and the inscription  $ins$  of net  $N$ , the behaviors of net  $N$  in Figure 2 can be described by the firing sequences that change the marking from one to the other.

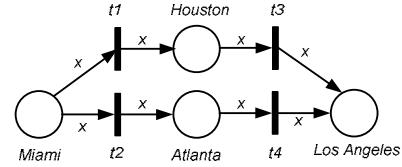


Figure 2. A PrT net structure

For example, the static and dynamic semantics of the net structure in Figure 2 can be formally specified as follows:

$$P = \{Miami, Houston, Atlanta, Los Angeles\}$$

$$T = \{t1, t2, t3, t4\}$$

$$F = \{(Miami, t1), (Miami, t2), (t1, Houston), (t2, Atlanta), (Houston, t3), (Atlanta, t4), (t3, Los Angeles), (t4, Los Angeles)\}$$

$$\varphi(Miami) = \varphi(Houston) = \varphi(Atlanta) = \varphi(Los Angeles) = \wp PERSON$$

$$R(t1) = x = 'John'; R(t2) = R(t3) = R(t4) = TRUE$$

// All places are defined to hold multiple tokens of the token type PERSON. Transition  $t1$  is enabled only when  $x = 'John'$ . There are no further constraints defined for transitions  $t2, t3$  and  $t4$ .

$$L(Miami, t1) = L(Miami, t2) = L(t1, Houston) = L(t2, Atlanta) = L(Houston, t3) = L(Atlanta, t4) = L(t3, Los Angeles) = L(t4, Los Angeles) = x$$

$$M_0(Miami) = \{< John >, < Mary >\}$$

$$M_0(Atlanta) = M_0(Houston) = M_0(Los Angeles) = \emptyset$$

The behavior of net  $N$  is the set of all execution sequences starting from the initial marking  $M_0$ . However, a possible firing sequence based on the above initial marking is  $M_0[t1>M_1[t2>M_2[t3>M_3[t4>M_4$  where the corresponding markings are as follows:

$$M_1(Houston) = \{< John >\}; M_1(Miami) = \{< Mary >\}$$

$$M_1(Atlanta) = M_1(Los Angeles) = \emptyset$$

$$M_2(Houston) = \{< John >\}; M_2(Atlanta) = \{< Mary >\}$$

$$M_2(Miami) = M_2(Los Angeles) = \emptyset$$

$$M_3(Los Angeles) = \{< John >\}; M_3(Atlanta) = \{< Mary >\}$$

$$M_3(Miami) = M_3(Houston) = \emptyset$$

$$M_4(Los Angeles) = \{< John >, < Mary >\}$$

$$M_4(Atlanta) = M_4(Houston) = M_4(Miami) = \emptyset$$

### 2.2 Modeling BDI Structure

*Beliefs* are the knowledge that an agent has about the world. Traditional approach in AI community is to represent knowledge symbolically as a collection of logical formulas [16] consisting of facts and rules. PrT nets are net representations of predicate logics and are amenable to represent logical sentences as a net structure [8, 17]. For example, a logical clause  $parent(x, y) \Rightarrow ancestor(x, y)$  can be represented by a net structure, in which a transition represents the implication, 'parent' as the input place and 'ancestor' as the output place of the transition respectively. The arc from place 'parent' to the transition and the arc from the transition to place 'ancestor' are labeled with the pair of variables  $\langle x, y \rangle$ .

We consider *desires* as the set of pre-defined agent goals that can be achieved with respect to the design objective of an agent model. Intuitively, agent goals are a set of reachable markings in a net with respect to some initial markings. Therefore, if a PrT net structure specifies the action model of an agent, then, *intentions* are the possible transition sequences that can reach goal markings from current markings. For example, in Figure 2, John is currently at Miami and intends to go to Los Angeles. Los Angeles is a goal. Nevertheless, there are two paths available from current location Miami to Los Angeles: (1) Miami-Houston-Los Angeles (2) Miami-Atlanta-Los Angeles. The transition sequences for the paths are  $M_0[t1>M_1[t3>M_2$  and  $M_0[t2>M_1[t4>M_2$ . The net structure in Figure 2 exhibits the non-determinism that addresses the autonomy of path selection to reach agent goal (Los Angeles).

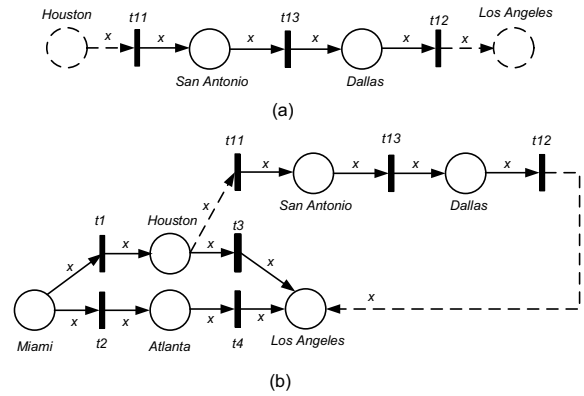
### 2.3 Modeling Aspects

Now, consider of choosing the path from current location Miami to Los Angeles, there are two viable paths to Los Angeles. However, there may exist a path of agent's best interest (e.g., choose whichever the fare is cheaper). To extend the action model in Figure 2 with reasoning behaviors, the net structure of agent knowledge can be modeled separately as a reusable aspect and woven at the point of making decision. In this case, the conclusion of a reasoning aspect can be woven at transition  $t1$  and  $t2$  as an additional enabling condition to enforce a desired path. Similarly, the action model of an agent is extendable for alternate plans, which can be modeled as aspects as well. As a consequence, the behavior adaptation of an agent model can be addressed. For example, if John travels by Houston, other traveling paths to Los Angeles from current location Houston may further be available. Thus, an aspect of alternate path can be woven at place 'Houston' to extend the adaptable behavior.

For aspect weaving, we borrow the terms from AspectJ [11]. First, a *join point* is a well-defined point, where additional behaviors can be woven into. A join point can be a transition or a place in a net structure. Second, an *advice* is a net structure that is to be woven into the associated net that has the specified join points. Third, a *pointcut* specifies the join points in associated nets. Fourth, an *aspect* is a modular unit that specifies *pointcuts* and *advices*. We consider *aspect weaving* as the process of connecting advices to the nets that are specified in pointcuts.

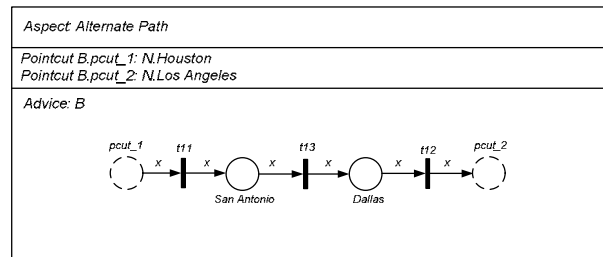
We demonstrate the aspect weaving of a net and an aspect using the net structure of Figure 2 in Section 2.1. Assuming there is an alternate path: Houston-San Antonio-Dallas-Los Angeles, and the net structure is shown in Figure 3(a). Place "Houston" in Figure 2 is considered as a *place join point* where alternate paths might be available. Figure 3(b) shows the woven net after appropriately weaving the aspect of an alternate path into the net in Figure 2 by connecting an incoming arc from "Houston" to

transition  $t_{11}$  and connecting an outgoing arc from transition  $t_{12}$  to place "Los Angeles".



**Figure 3. (a) An alternate path (b) A woven net.**

In order to properly weave different nets together, we need to specify the aspect and its join point(s). Thus, the specification of an aspect should include (1) the name of the aspect (2) pointcuts, which specify the connecting points of relevant nets (3) advice, which is a net structure to be woven. As a result, the alternate path aspect (Figure 3(a)) with respect to the original net in Figure 2 can be specified as shown in Figure 4, where the name of original net is represented by  $N$ , and the name of advice is represented by  $B$ . The *pointcuts* defined in the aspect 'Alternate Path' specify the join points of net  $N$  in a format as follows: (*advice\_name.pointcut\_name: net\_name.join\_point1 [,net\_name.join\_point2, ...]*).

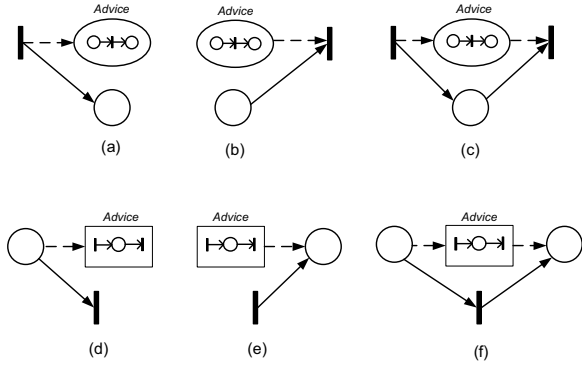


**Figure 4. An alternate path aspect.**

The above example shows that there are two possible kinds of join points in a net, namely transition join point and place join point. A place join point is considered as a place where an aspect of alternate choice can be added; or, as a place that can hold the tokens generated from an aspect of some extended behaviors. A transition join point is considered as a point where an aspect of some concurrent behaviors can be added; or, where an aspect of additional enabling conditions can be added. We generalize some weaving patterns in addition to previous example and show them in Figure 5, where (a), (b) and (c) are the patterns of *transition join point* since the weaving point is at a *transition*; and, the patterns in (d), (e) and (f) are *place join point* since the weaving point is at a *place*. Patterns (a) and (d) are similar to *after advice* in AOP; (b) and (e) are



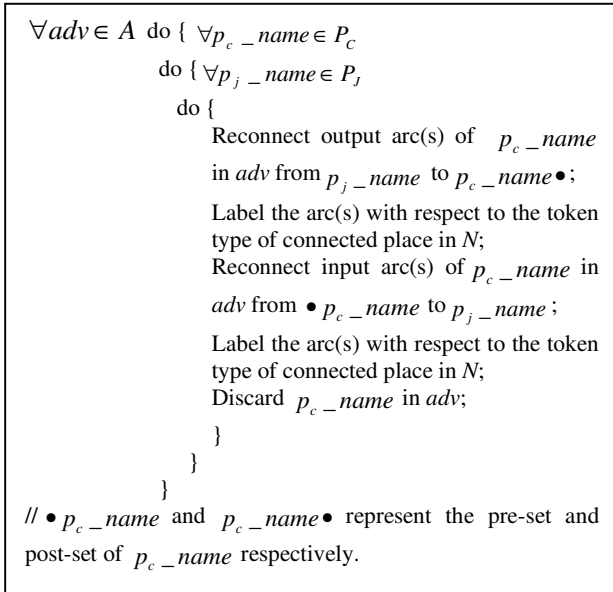
similar to *before advice*; (c) and (f) are similar to *around advice*, which add an explicit control to a net.



**Figure 5. General weaving patterns.**

Intuitively, during a weaving process, a transition join point must be connected with a place in the advice and a place join point must be connected with a transition in the advice. This is to ensure the correctness of the syntax and static semantics of a woven net. The weaving rules for both transition and place join point(s) are as follows.

- (1) For every join points that are specified in the pointcut(s) of an aspect, reconnect the input arc(s) and/or output arc(s) that are associated with the specified pointcut(s) in the advice with the join points in original net.
- (2) Label the arc(s) with the same type of variable(s) with respect to the join point(s) in original net.
- (3) The dotted places and transitions that highlight the specified pointcuts in advices are discarded after weaving process.



**Figure 6. The weaving process of aspects and associated nets.**

Let  $N$  be the net structure (desires) that represent basic agent plan, and  $A$  be the set of all aspects, which may include the reasoning behavior (beliefs) and alternate plans

(intentions). For all aspects in  $A$  such that each advice  $adv$  has a set of *pointcuts*  $P_C$ , in which each pointcut  $p_c\_name$  specifies the join point in advice  $adv$  and a set of associated join points  $P_J$  in net  $N$  in a form of  $adv.p_c\_name.: N.p_j\_name.[N.p_{j+1\_name}, \dots, N.p_{n\_name}]$  in order to compose  $A$  and  $N$ . A pointcut  $p_c\_name$  could be a transition or a place; however, the associated pair of  $p_c\_name$  in  $P_C$  and the join points  $p_j\_name$  in  $P_J$  must be of both transitions or places. The composition of  $A$  and  $N$  is through a weaving process that is briefly described in Figure 6.

### 3. Related Works

On formal approach for modeling multi-agent systems, there were works based on temporal logic [2, 19, 22] and Z notation [15]. Although temporal logic is well suited in specifying the properties of reactive agents based on BDI agent architecture [19], its property-oriented nature provides no state transition relations. Thus, it is difficult to map a temporal logic specification to an implementation. Z notation provides rich type definitions. However, Z notation provides no explicit operational semantics and no effective definition of the concurrency, thus is insufficient to specify the concurrent and interacting behaviors of agents. Furthermore, in [15], there was no discussion about BDI structure and rational behaviors.

Among Petri net based research works for modeling agents, most of the works were focused on the control structure and the mobility of agents (e.g., the works in [13, 23]). In [25], BDI model was used in their framework based on object-oriented Petri nets; however, the focus was on the modeling of message passing and a planner module where BDI was represented by places. In [14], a component-based modeling approach was developed based on Colored Petri nets by the invention of potential arcs to address the resource conflict resolution among agents. The focus was on the construction of agent plans that were free of conflicts. In [17], a mechanism has been derived to transform a logic program that is represented by a set of Horn clauses to an equivalent PrT net structure. The logical sentences were represented as a net structure in [17].

To the best of our knowledge, there was no work integrating aspect-oriented concept with PrT nets for modeling multi-agent systems based on BDI structure. In [6], Unified Modeling Language (UML) was used to model the aspectual components in multi-agent systems. In [24, 26], security concern was modeled individually based on PrT nets and woven into a base net to generate a secured net model. Nevertheless, both of the works [24, 26] were not related to multi-agent system modeling and limited to security aspect.

### 4. Concluding Remarks

We address the essential concerns of modeling an agent by adopting BDI structure, and the structural complexity of modeling a BDI agent by incorporating aspect-oriented concept into PrT nets. Our modeling approach enhances the modularity and adaptability of PrT net models. As a result,

net models are more manageable to adapt different concerns for the analysis of critical aspects in multi-agent systems. This paper presents a conceptual model of our ongoing research for modeling multi-agent systems using PrT nets. The future works of our study include the detailed modeling approach of essential aspects; such as modeling the reasoning aspect where the rules can be dynamically changed and applied instead of modeled as a static net structure.

**Acknowledgements.** This work was partially supported by NSF grants HRD-0833093 and IIP - 0738465.

## Reference

- [1] G. Argote, P. Clarke, X. He, Y. Fu, and L. Shi: "A Formal Approach for Translating a SAM Architecture to PROMELA", Proc. of the International Conference on Software Engineering and Knowledge Engineering (SEKE08), San Francisco, July, 2008.
- [2] H. Barringer, M. Fisher, D. Gabbay, G. Gough and R. Owens, METATEM: A Framework for Programming in Temporal Logic, Proceedings on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness., LNCS, Vol. 430, pp.94-129.
- [3] L. Chang, J. Ding, X. He, S. Shatz: A Formal Approach for Modeling Software Agents Coordination, Communication of SIWN, Vol. 3, 2008, pp.58-64.
- [4] K. Fischer, J. P. Muller, M. Pischel: A Pragmatic BDI Architecture, Intelligent Agent II, Vol. 1037, pp.203-218, Springer-Verlag, 1995.
- [5] Y. Fu, Z. Dong, and X. He: "A Translator of Software Architecture Design from SAM to Java". International Journal of Software Engineering and Knowledge Engineering, vol. 17, no.6, 2007, 709-755.
- [6] A. Garcia, U. Kulesza, C. Lucena: Aspectizing Multi-agent Systems: From Architecture to Implementation, SELMAS, LNCS Vol. 3390, pp.121-143, February 2005, Springer Berlin / Heidelberg.
- [7] H. J. Genrich, Predicate/Transition nets. Advances in Petri Nets 1986, pp. 207-247.
- [8] X. He, W. C. Chu, H. Yang: A New Approach to Verify Rule-Based Systems Using Petri Nets, Information and software Technology, Vol. 45, No.10, pp.663-669, 2003.
- [9] N. Jennings and M. Wooldridge, Agent-Oriented Software Engineering. Proceedings of the 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, 2000.
- [10] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J. M. Loingtier, J. Irwin: Aspect-oriented programming. In Proceedings of the European Conference on Object-Oriented Programming (ECOOP). Springer-Verlag LNCS 1241, June 1997.
- [11] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W. Griswold: Getting Started with AspectJ, Communication of ACM, Vol. 44, No. 10, pp. 59-65, 2001.
- [12] D. Kinny, M. Georgeff and A. Rao, "A Methodology and Modeling Technique for Systems of BDI Agents," Proceedings of the Seventh European Workshop on Modeling Autonomous Agents in a Multi-Agent World, 1996.
- [13] M. Kohler, H. Rolke, Modeling Mobility and Mobile Agents Using Nets Within Nets, Proc. of International Conf. on Application and Theory of Petri Nets, LNCS vol. 2679 (2003), 121-139.
- [14] J. Lian and S. M Shatz, Potential arc: A Modeling Mechanism for Conflict Control in Multi-agent Systems. Proceedings of the 4th Symposium on Design, Analysis, and Simulation of Distributed Systems (DASD-06) 2006, pp. 467-474.
- [15] M. Luck, N. Griffiths and M d'Inverno, From Agent Theory to Agent Construction: A Case Study. Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III 1997, Vol. 1193, Springer-Verlag: Heidelberg, Germany, pp. 49-64.
- [16] J. McCarthy: Programs with Common Sense, Semantic Information Processing, pp. 403-418. Cambridge, MA, MIT Press.
- [17] T. Murata, D. Zhang: A Predicate-Transition Net Model for Parallel Interpretation of Logic Programs, IEEE Transactions on Software Engineering, Vol. 14, No. 4, 1988..
- [18] A. S. Rao, M. Georgeff.: BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pages 312-319, San Francisco, CA, June 1995.
- [19] A. S. Rao, M. Georgeff: Modeling Rational Agents within a BDI Architecture, Proceedings of Knowledge Representation and Reasoning, pp.473-484, 1991.
- [20] K. Sycara, J. A. Giampapa, B. Langley, M. Paolucci: The RETSINA MAS, a Case Study, in SELMAS 2003, pp.232-250. Springer-Verlag.
- [21] M. Wooldridge, "An Introduction to Multi-agent Systems", J. Wiley, New York, 2002.
- [22] M. Wooldridge. Reasoning about Rational Agents. The MIT Press: Cambridge, MA, 2000.
- [23] D. Xu, J. Yin, Y. Deng and J. Ding: A Formal Architecture Model for Logical Agent Mobility. IEEE Transactions on Software Engineering. Vol. 29, No. 1, pp. 31-45, Jan. 2003.
- [24] D. Xu, K. E. Nygard: Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets. IEEE Transactions on Software Engineering, Vol.32, No.4, pp.265-278, IEEE Press.
- [25] H. Xu and S. M. Shatz, A Framework for Model-based Design of Agent-oriented Software. IEEE Transactions on Software Engineering, 2003, pp. 15-30.
- [26] H. Yu, D. Liu, X. He, L. Yang, S. Gao: Secure Software Architectures Design by Aspect Orientation, Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems, pp.47-55, 2005, IEEE Computer Society.

# A Multi-Agent Debugging Extension Architecture

Ziad Al-Sharif, Clinton Jeffery  
Computer Science Department  
University of Idaho  
zsharif@ieee.org, jeffery@cs.uidaho.edu

## Abstract

*The Idaho Debugging Extension Architecture (IDEA) enables dynamic analysis agents, such as automatic debugging and visualization agents, to be loaded on the fly in a source-level debugger. IDEA is an event-driven debugging architecture that provides a simple interface to load API-compliant external dynamic analysis agents during a debugging session. Multiple standalone agents can be loaded and managed under the control of the source-level debugger. Successful agents can be migrated into the source code of the debugger core as permanent features with higher performance.*

## 1. Introduction

A source-level debugger helps programmers locate bugs by stepping through the source code and examining the current state of the execution. Some drawbacks of typical source-level debuggers are: 1) limited information provided about the execution history, 2) lack of automated, analysis-based debugging techniques, and 3) closed architecture that provides little or no cooperation with external debugging and visualization tools.

Reversible and post-mortem (trace-based) debuggers, such as ODB[13], TOD[15], and Whyline [11,12] provide debugging techniques based on the ability to browse forward and backward through the states of an execution. This approach provides outstanding debugging capabilities such as finding where and why some action has happened, but poses formidable scalability problems, and is good at finding some types of bugs and not others. While they provide valuable capabilities, some trace-based debuggers neglect common debugging techniques such as altering the state of the buggy program.

This paper presents the Idaho Debugging Extension Architecture (IDEA). IDEA supports extensions called *agents*. An agent is an event-driven task-oriented program execution monitor. IDEA's agents are written and tested as standalone programs, after which they can be loaded and used on the fly from within the conventional source-level debugger, or integrated as permanent features into the debugging core, with almost no source code alteration. Different agents perform different debugging missions such as detecting a suspicious execution behavior, performing an

automatic debugging procedure, or executing a dynamic analysis technique.

IDEA's debugging extension agents monitor the execution of a program for specific run time events; an *event* is an action during the execution of the program such as a method being called or a major syntax construct being entered. Different agents can be loaded and active, and each agent receives different runtime events based on their own request. Agents are coordinated by a central debugging core. Each agent 1) provides the debugging core with its set of desired events, 2) receives relevant events from the debugging core, 3) performs its debugging mission, which may utilize execution history prior to the current execution state, and 4) sends its analysis results back to the user.

IDEA processes and filters execution events, manages the debugging session, and handles external and internal debugging agents. IDEA allows the debugging core and any number of compatible dynamic analysis agents to assist in locating and finding bugs. Separately-compiled dynamically-loaded external agents receive their information from IDEA's debugging core, which controls them. All active external/internal agents are suspended whenever a breakpoint or a watchpoint is reached, and they are resumed whenever the user resumes the buggy program. The external debugging agents' standard inputs and outputs are redirected and coordinated by IDEA's debugging core.

## 2. Debugging with Agents

Conventional debuggers allow users to explore their debugging hypotheses using manual investigation. Debugging with agents leverages the conventional debugging process by empowering the user with more tools to inspect the state of the buggy program. IDEA's agents may retain information beyond the current state of execution and perform automatic debugging and dynamic analysis techniques that could be supported by trace-based debuggers such as ODB [13,15]. However, IDEA's agents are task-oriented; each agent embodies a lightweight task-specific analysis technique.

IDEA's agents can be written and tested as standalone programs and then loaded into the debugger to work in concert with each other. Using IDEA, it is easy to define debugging agents that capture specific execution behaviors

such as: 1) loops that iterate  $N$  times, for some  $N \geq 0$ ; 2) variables that are read and never assigned or assigned and never read *during a particular execution*; 3) expressions such as subscripts that fail silently in a context where failure is not being checked; 4) a variable that may change its type during the course of execution; or 5) a trace of variable states, which allows users to trace backward and see where a specific variable was assigned long before it is involved in a crash. For example, many functions return a specific value when they encounter an error or fail to accomplish their job. An agent can automatically catch any of these failed functions and save the user the time that can be spent during a manual inspection.

Furthermore, IDEA's agents are employed within the conventional source-level debugging session, which provides a simple interface to load, unload, enable, or disable debugging agents on the fly, and the user can be selective about which agent(s) to use.

### 3. Design

IDEA features novel properties that distinguish it from other debugging architectures. First, it provides two types of extensions: dynamic extension on the fly during the debugging session (external agents), and formal steps for migrating and adopting standalone agents as permanent debugging features (internal agents). Second, it encourages users to write their own agents and incorporate them into a typical source-level debugging session. Finally, it supports an interactive users interface, where simultaneous agents can be loaded and managed during a debugging session. The user does not need to restart the debugging session whenever a decision is made to incorporate any of the debugging agents in that session. In contrast, common static and dynamic analysis tools and libraries have to be linked in advance into the source code of the buggy program, or initialized at the start of the host debugger.

IDEA's debugging core is comprised of five major components: 1) a console that provides the interface between the user and the debugging facilities, 2) a session that initializes and coordinates the debugging situation, 3) a debugging evaluator that provides the main monitoring loop and event filtering, 4) an agents interface that facilitates and provides the programming interface for external and internal extensions, and 5) a debugging state that maintains and shares the state of the debugger between the rest of the components and the user. See Figure 1.

### 5. Implementation

IDEA's implementation is based on two components that make the source-level debugger an event coordinator for the extensions; **Internals** and **Externals**. These components are plugged in to the main debugging loop as extra listeners on the runtime events. IDEA manages and coordinates the

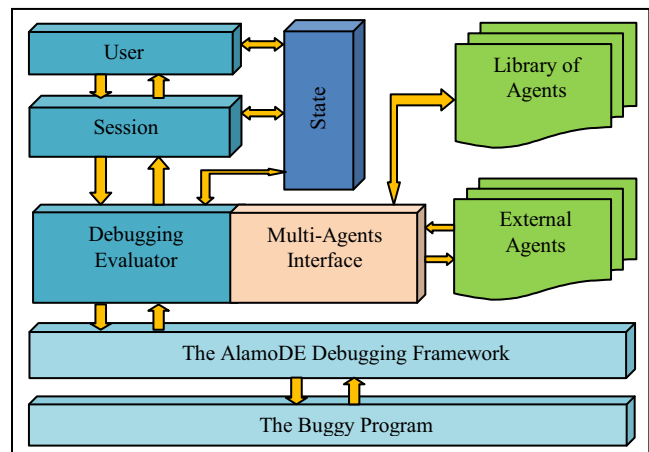


Figure 1. The IDEA Architecture

external agents and forwards received events, from the buggy program into different external debugging agents based on their interest. Extensions to an IDEA-based debugger are coordinated within the agents' interface of the evaluator component. Extensions are abstracted by objects which serve as Proxies for external agents or as Listeners in the case of internal extensions.

The Alamo monitoring framework provides high level primitives to control the buggy program and to customize the reported events. The next reported event will be one of those specified by a set of event types called an *event mask*; a detailed event filter for the current set of desired events. For the selected event types, if the *event code* has a corresponding entry in the hash table named *value mask*, then only those events that have a matched value is reported. This optimization limits the number of reported events to the inquired ones, and reduces the number of context switches.

IDEA's debugging core coordinates all of the built-in classical debugging techniques, the internal agents, and the dynamically loaded external agents. For every received event, first it checks whether any classical action is needed such as a breakpoint, or watchpoint. Then it checks for any enabled internal and/or external agent; it forwards events to the enabled agents based on their event mask.

### 6. Extensions

IDEA supports two types of agents: 1) standalone agents that can be loaded on the fly, from any point, during the debugging session, and 2) built-in debugging agents that are incorporated into the debugging core as permanent debugging features.

The event masks of extension agents (internals and externals) are added to the set of events that are requested by default by the debugging core. On the fly, the debugging core starts asking the buggy program about those extra events. When the debugging core receives an event from the buggy program, it forwards the received event to those extension agents that are enabled and requested this event in

their event mask. For internal agents, this takes the form of a call to a listener method, while for external agents it takes the form of a thread switch (Unicon threads are called *co-expression*), which the agent sees as a return from its `EvGet()` event request. `EvGet()` is an Alamo primitive that resumes the buggy program until the next available event. See Figure 3.

Different agents can be loaded and active, and each agent receives different runtime events based on their own event mask. An extension agent may change its event mask during the course of execution. A change on any extension agent's event mask immediately triggers an update of the event mask of the debugging core and alters the set of events received by the debugging core and forwarded to the extension agent.

### 6.1. Sample Agent

The code provided in Figure 2 shows a prototype of an IDEA-based agent. It is a toy example that captures the number of calls of user-defined functions/methods and native built-in functions, and finds the ratio for each call type. This provides a rough measure of the degree of VM overhead for a particular application. The class `Example()` contains three types of methods: 1) event *handlers*, which collect information based on the received events; handler methods start with the prefix *“handle\_”* followed by the name of the event code (`handle_E_Pcall()`), 2) information *analyzers*, which analyze the collected information by the event handlers; analyzer methods start with the prefix *“analyze\_”* followed by any name (`analyze_info()`), and 3) information or result *writers*, which output the result found by the agent; writer methods start with the prefix *“write\_”* followed by any name (`write_info()`). Agents that follow this method naming convention can be registered automatically with the library of internal agents. Otherwise, agents can be registered manually. See Section 6.4.

### 6.2. External Agents

External agents can be written and tested as standalone tools, and subsequently loaded on the fly and used together during a debugging session. IDEA's external agents are loaded and controlled by its debugging core. Active agents are paused whenever the buggy program is paused and they resume whenever it resumes.

IDEA's debugging core receives runtime events from the buggy program based on the current debugging context, and the event masks of the external agents. The `Externals` component multiplexes the received events between different external agents. Events are sent to related active agents. An external agent requests events from the debugging core using the `EvGet()` primitive, which transfers control from the external agent to the debugging core. `EvGet()` is the same primitive that transfers control

```

$include "evdefs.icn"
link evinit
class Example(eventMask, pcalls, fcalls, prate, frate)
method handle_E_Pcall()
    pcalls += 1
end
method handle_E_Fcall()
    fcalls += 1
end
method analyze_info()
    total := pcalls + fcalls
    prate := pcalls / total * 100
    frate := fcalls / total * 100
end
method write_info()
    write(" # pcalls = ", pcalls, " at rate :", prate)
    write(" # fcalls = ", fcalls, " at ratio :", frate)
end
initially()
    eventMask := cset(E_Pcall || E_Fcall)
    pcalls := fcalls := 0.0
end
procedure main(args)
    EvInit(args)
    obj := Example()
    while EvGet(obj.eventMask) do
        case &eventcode of {
            E_Pcall: { obj.handle_E_Pcall() }
            E_Fcall: { obj.handle_E_Fcall() }
        }
        obj.analyze_info(); obj.write_info()
    end
end

```

Figure 2. An IDEA-based agent prototype

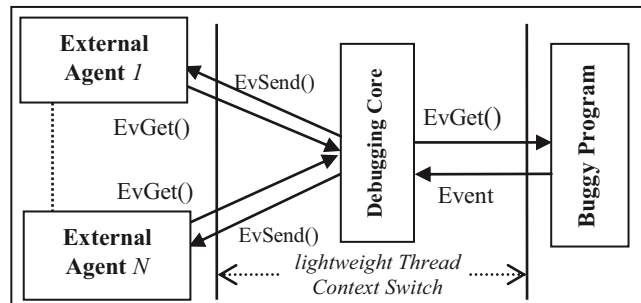


Figure 3. IDEA's general control/events flow

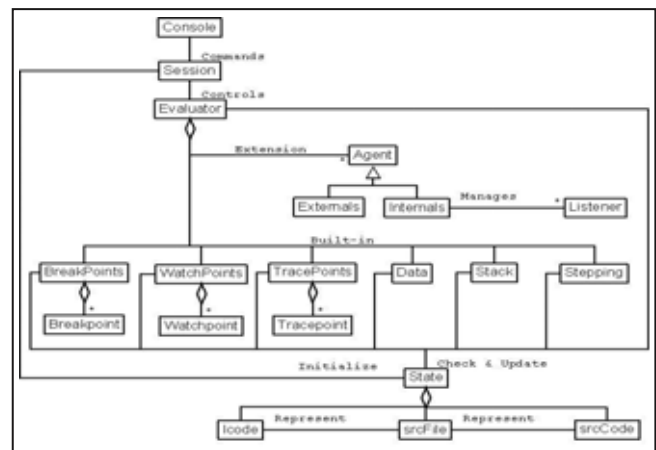


Figure 4. UML of UDB; An IDEA-based debugger

and acquires events from the buggy program when the agent is used in a standalone mode. The **Externals** component forwards events to any of the external agents using the `EvSend()` primitive, which is another Alamo primitive that sends the last event received by the debugging core to the external agent. A context switch occurs whenever control transfers between the debugging core and either a buggy program or an external agent. Event forwarding is accomplished without the knowledge of the external agent itself, which means the external agent needs no modification to be loaded and used by IDEA's core.

### 6.3. Internal Agents

Besides support for whole programs as external agents, IDEA supports insertion of dynamic analyses into the debugging core as a listener agent that implements a set of callback methods. IDEA's debugging core implements different built-in agents for different classes of bugs. For performance reasons, each agent has its own implementation based on the type and the combination of events that the debugging core must monitor in the buggy program.

The **Internals** component handles the built-in agents. Internal agents are called from the main debugging loop with a call to the `forward()` method of the **Internals** component, where internal agents are registered during initialization. The **Internals** component checks which agents are active and calls the related underlying method(s) based on the event code that is received by the debugging core.

### 6.4. Migration from Externals to Internals

External agents allow automatic debugging techniques based on various dynamic analyses to be developed and tested easily in the production environment. Selected external agents may become internal—built-in monitors within the debugging core for improved performance. Internal agents do not pay the (lightweight, but still painful) cost of the context-switch communication between the debugging core and the external agents. IDEA provides smooth migration from external agents to internal. The first issue in migration is to accept a callback-style event listener architecture in place of the more general `main()` procedure that an external agent uses from a separate thread. IDEA provides an abstract class called **Listener**, which must be subclassed within the external agent before the external can be used as internal. The **Listener** class allows the debugging core to acquire the event mask of the migrated internal agents, and to determine which listener methods to use for the various event types.

The agent prototype discussed in Figure 2 can be used as a standalone program or as an external agent under IDEA without any modification. In order to move such an external agent to an internal one, the user must derive this **Example**

class from IDEA's **Listener** abstract class and register it in the **Internals** class. Whenever its own event mask changes, this abstract class helps the **Internals** class rebuild the event mask for the internal agents and the debugging core by calling the `updateEventMask()` method in IDEA's **State** class to update the debugging core with the new event mask obtained from the internal agent.

An object of the newly migrated internal agent must be instantiated and inserted into the list of clients in the **Internals** class. This can be done through the method `register()` from the **Internals** class. For example, to register the prototype **Example** agent provided Figure 2 as an internal agent, the programmer has to place a call to the method `register()` in the constructor of the **Internals** class where the first parameter associates the agent with a formal name as an ID during the debugging session, and the second parameter is an object of that agent class (`register("call_count", Example())`). This is the simple automatic registration that applies for agents who follow the sample agent convention shown in Figure 2 and discussed in Section 6.1. To register a complex agent that does not follow this sample convention, the method `register()` can be called with three extra parameters to register the method *handlers*, the *analyzers*, and the *writers* respectively.

```
register( "call_count" , Example() ,  
        ["handle_E_Pcall", "handle_E_Fcall"] ,  
        ["analyze_Info"], ["writer_Info"] )
```

Furthermore, the new internal agent must be stripped of its `main()` procedure before compilation and linking into the debugging core. Alamo primitives found in the external agent are no longer needed when it becomes an internal agent. `EvInit()` is needed once per each buggy program and it is already performed by the debugging core. IDEA's `EvGet()` asks the buggy program for the next event, so the extension agent has no need to call this function. IDEA forwards events to relevant enabled internal agents based on their event mask. The mapping of events such as `E_Pcall` to their listener methods (`handle_E_Pcall`) is constructed automatically and used by the **Internals** class `report()` method.

## 7. Evaluation

In order to evaluate and refine the IDEA architecture, a debugger called UDB was constructed. UDB is an event-driven source-level debugger for Icon and Unicon programs. It implements the classical debugging features found in a typical debugger such as GDB, and it utilizes IDEA's agent-based extensions. See Figure 4.

An IDEA-based debugger must use different approaches to implement features found in standard source-level debuggers, and faces potential performance challenges. In compensation, this type of implementation greatly simplifies the process of experimenting with new debugging

techniques that probably would not be undertaken if the implementation was limited to the low-level approaches found in other debuggers.

Nevertheless, one of the biggest considerations in the design of a source-level debugger is the performance. In IDEA, a considerable amount of time is spent on: 1) processing the instrumentation in the buggy program, 2) filtering the received events in the debugging core, and 3) processing the context switches between the debugging core, the buggy program and the external debugging agent. Since IDEA's debugging core is a mediator between those external agents and the buggy program, each external agent imposes two extra context switches. For events that the debugging core does not itself need, there are two levels of context switches where outside IDEA there would be only one: the first is between the debugging core and the buggy program, and the second is between the debugging core and the external agent. Alamo's event filtering techniques, provided by the event mask and the value mask, reduce the amount of context switches; a context switch occurs only when there is an event and it is needed by either the debugging core or any of its agents.

Under UDB, eight different debugging agents were loaded and tested as external agents, and then migrated to become part of the UDB's library of internal agents. The slowdown imposed by the external agents was at most 3 times slower than the standalone agent mode, and the slowdown imposed by the migrated internal agents, was at most 2 times slower than the standalone agents. This suite of debugging agents imposes at most 20 times slowdown on the execution of the buggy program over an uninstrumented execution mode, but in the general case, the slowdown depends on the algorithms used by the dynamic analysis technique implemented by the debugging agent. To place this in perspective, a debugger such as *valgrind* [14] imposes a 20 to 50 times slowdown, and it does not provide the interactive debugging environment that IDEA and its debugging tools provide, where the user can be selective about which and where to enable/disable agents from within a breakpoint based debugging session.

## 8. Related Work

Standard source-level debuggers such as GDB [16] and its graphical front end DDD [20] provide convenient debugging and tracing facilities. But using a conventional source-level debugger is still time consuming; it is largely based on forming a hypothesis and guessing where to place breakpoints. One way to reduce the debugging time is to automate the debugging process. Automated debugging is a challenging problem that goes back to the 60's [5] and mid 70's [10]. Its most challenging part is the reasoning about the information supplied by the debugging tool, which still heavily depends on the human factor.

Trace-based debuggers such as ODB [13, 15], provide debugging facilities by tracing the complete program

history of states. In ODB, the program must be traced first before the debugging techniques can be used, but it provides the ability to investigate backward in the execution history. IBM's JInsight [3] combines tracing and visualization techniques, but it generates a huge amount of traced data in a short period of time. In general, the most common problem of trace-based debuggers is the huge amount of traced data that limits the scalability and raises the level of complexity. An example of a recent tool is JDLab [1], which applies the graph theoretical algorithms to reduce the amount of traced events.

A debugging architecture such as JPDA, with its lowest level JVM TI [6], provides an event-based debugging infrastructure and enough events for conventional debugging, profiling, and visualization. JVM TI supports about thirty five kinds of events, whereas IDEA incorporates more than one hundred kinds of events. IDEA users use events, event-sequences, and event-patterns to write their own debugging agents that detect specific execution behaviors— some of which are suspicious behaviors and others are defined bugs.

Both IDEA and JVM TI, provide techniques to inspect the state and to control the buggy program running in the VM. JVM TI agents must be loaded and initialized at the start of the JVM, whereas IDEA's extension agents can be loaded on the fly from any point during the debugging session. IDEA's debugging agents can be designed and tested as standalone programs, then dynamically linked into debugging sessions, and finally incorporated in the debugger source code as permanent extension.

*Valgrind* [14] provides a formal mechanism for custom extensions, but it does not provide dynamic extensions or interactive debugging. IDEA provides a debugging architecture that maintains the interactive debugging found in conventional debuggers such as GDB, and it provides enough events to maintain lightweight task-oriented custom trace-based debugging agents.

## 9. Future Work

Our future work aims at using IDEA to develop a wide range of automatic debugging agents. Performance can be further improved by buffering related events and avoiding extra context switches; this will help for both internal and external extension agents. Another potential improvement is to offload the cost of external agents onto additional processor cores. The simplest speedup approach initially will be to add the most useful agents of automatic debugging techniques as internals, which will reduce the context switches on external agents.

Subsets of the Alamo framework used by IDEA for Unicon debugging have been implemented for monitoring C and Python [9, 19]. Future work may extend IDEA's debugging facilities to these languages, or port IDEA to run on other debugging platforms such as JPDA. Another potential future work is to use an instrumentation

framework, such as ASM, PIN, and Atom, as a substitute for Alamo.

## 10. Conclusion

Different programmers and bugs require different debugging techniques. It is impossible to provide every desirable debugging technique in one tool. IDEA provides a compromise solution by allowing programmers to run a chosen suite of dynamic analysis agents from within a normal interactive debugging session. The IDEA architecture: 1) combines the capabilities of classical and trace-based debuggers, 2) lets users write their own high level standalone debugging agents and loads them from any point into an interactive debugging session, and 3) provides simple mechanism to incorporate standalone programs as internal permanent features.

Compared with the slowdown of many automatic debugging techniques, the performance of IDEA is very good. However, the true test of IDEA's performance will be whether it enables debugging agents that justify their time cost by the value they provide to programmers, as valgrind often does.

## 11. Acknowledgments

This research was supported in part by an appointment to the National Library of Medicine Research Participation Program. This program is administered by the Oak Ridge Institute for Science and Education for the National Library of Medicine.

## 12. References

- [1] Alekseev, S. 2007. Java debugging laboratory for automatic generation and analysis of trace data. In *Proceedings of the 25th Conference on IASTED international Multi-Conference: Software Engineering* (Innsbruck, Austria, February 13 - 15, 2007). W. Hasselbring, Ed. ACTA Press, Anaheim, CA, 177-182.
- [2] Bruneton, E. A Java bytecode engineering library, ASM 3.0, Feb 2007, <http://asm.objectweb.org>.
- [3] De Pauw, W., Jensen, E., Mitchell, N., Sevitsky, G., Vlissides, J., and Yang, J., *Software Visualization, State-of-the-Art Survey*. LNCS 2269, Stephan Diehl (ed.), Springer Verlag, 2002.
- [4] Griswold, R. E., and Griswold, M. T. 1997. The Icon Programming Language. Peer-to-Peer Communications, Inc., San Jose, California.
- [5] Jacoby, K. and Layton, H. 1961. Automation of program debugging. In *Proceedings of the 1961 16th ACM National Meeting* ACM, New York, NY, 123.201-123.204.
- [6] Java Platform Debugging Architecture, <http://java.sun.com/javase/6/docs/technotes/guides/jpd>.
- [7] Jeffery, C. L., 1999. Program Monitoring and Visualization: an Exploratory Approach, Springer New York.
- [8] Jeffery, C. L., Mohamed, S., Pereda, R., and Parlett, R. 2004. Programming with Unicon. <http://unicon.org/book/ub.pdf>.
- [9] Jeffery, C., Zhou, W., Templer, K., and Brazell, M. 1998. A lightweight architecture for program execution monitoring. *SIGPLAN Not.* 33, 7 (Jul. 1998), 67-74.
- [10] Katz, S. and Manna, Z. 1975. Towards automatic debugging of programs. In *Proceedings of the international Conference on Reliable Software* (Los Angeles, California, April 21 - 23, 1975). ACM, New York, NY, 143-155.
- [11] Ko, A. 2006. Debugging by asking questions about program output. In *Proceedings of the 28th international Conference on Software Engineering* (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM, New York, NY, 989-992.
- [12] Ko, A. J. and Myers, B. A. 2004. Designing the whyline: a debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, New York, NY, 151-158.
- [13] Lewis, B. *Debugging Backward in Time*. Proceedings of the Fifth International Workshop on Automated Debugging. AADEBUG 2003, September 2003, Ghent.
- [14] Nethercote, N. and Seward, J. *Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation*. Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, San Diego.
- [15] Pothier, G., Tanter, É., and Piquet, J. 2007. Scalable omniscient debugging. In *Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object Oriented Programming Systems and Applications* (Montreal, Quebec, Canada, October 21 - 25, 2007). OOPSLA '07. ACM, New York, NY, 535-552.
- [16] Stallman, R. M., Pesch, R., Shebs, S., et al. 2002. *Debugging with GDB: the GNU Source Level Debugger*. <http://sourceware.org/gdb/documentation>.
- [17] Templer, K. and Jeffery, C. 1998. A Configurable Automatic Instrumentation Tool for ANSI C. In *Proceedings of the 13th IEEE international Conference on Automated Software Engineering* (October 13 - 16, 1998). Automated Software Engineering. IEEE Computer Society, Washington, DC, 249.
- [18] Tzoref, R., Ur, S., and Yom-Tov, E. 2007. Instrumenting where it hurts: an automatic concurrent debugging technique. In *Proceedings of the 2007 international Symposium on Software Testing and Analysis* (London, United Kingdom, July 09 - 12, 2007). ISSTA '07. ACM, New York, NY, 27-38.
- [19] Zeller, A. and Lütkehaus, D. 1996. DDD—a free graphical front-end for UNIX debuggers. *SIGPLAN Not.* 31, 1 (Jan. 1996), 22-27.
- [20] Ziad Al-Sharif. *Debugging with UDB, User Guide and Reference Manual*. Unicon Technical Report #10.



# A Recognition-Primed Architecture for Human-Centric Multi-Agent Systems

Xiaocong Fan  
The Behrend College  
The Pennsylvania State University  
Erie, PA 16563, USA  
xfan@psu.edu

## Abstract

*Human-centric multi-agent systems in dynamic domains often involve a great level of human-agent collaboration where time stress and high stakes are the norm. Drawing upon Klein’s recognition-primed decision model, we propose a recognition-primed agent architecture for building human-centric multi-agent systems. The concept of timed Petri nets is extended to explicitly capture the timing constraints and collaboration points along teamwork processes where both human and agents can play a role. In addition, we describe how the idea of recognition-primed agent architecture is implemented in SMMall—a framework for developing human-centric cognitive agents.*

## 1. Motivation

Agent architecture has been one of the key research areas of multi-agent systems. Many architectures have been proposed, investigated, and successfully applied in various domains. For example, Soar [7, 12] and ACT-R [1] are two notable cognitive agent architectures, STEAM [15] and CAST [17] are architectures that focus on cooperation and collaboration among intelligent agents.

Recently, researchers in multi-agent systems field have manifested increasing interests in using intelligent agents to model, simulate, and support human teamwork behaviors [9, 4, 2]. However, only a few focused on agent architectures for human-centric systems. For instance, Norling examined how to extend the well-known BDI architecture with Klein’s recognition-primed decision (RPD) model to support human-like decision making [9]. Interestingly, the CAST [17] system was also extended with a computational RPD model and resulted in the R-CAST [2].

The objective of this research is three-fold. We first would like to argue that Klein’s RPD model itself actually can be employed as a generic cognitive agent architecture; the RPD model can be treated as a reification of the stan-

dard sense/decide/act loop. Second, because decision making teams in time-stressed domains rely more and more on the timeliness and quality of decision making activities, we choose to use extended timed Petri nets to explicitly capture timing constraints of a teamwork process. Lastly, as a proof-of-concept, we briefly describe how the generic architecture is implemented in SMMall—a framework for developing human-centric cognitive agents.

The rest of the paper is organized as follows. Relevant research background is introduced in Section 2. In Section 3, we extend the timed Petri-net to define collaboration net, and present a formalism of a team RPD process. In Section 4, we describe how the generic architecture is implemented in SMMall. Section 5 concludes the paper.

## 2. Research Background

Klein’s Recognition-Primed Decision framework (RPD) [6] is a naturalistic decision making model that attempts to capture how human experts make decisions in realistic settings based on the recognition of similar experiences.

The original RPD model includes a recognition phase and an evaluation phase. In the recognition phase, a decision maker synthesizes the observed features about the current decision situation into appropriate cues or pattern of cues, then uses a strategy called “feature-matching” to recall similar cases by matching the synthesized cues with previous experience. In the case that feature-matching cannot provide an acceptable solution due to lack of information or experience, another strategy called “story-building” is used to develop a potential explanation of how the current situation might have been emerging, and a workable solution can be suggested afterward.

The recognition phase has four products: relevant cues (what to pay attention to), plausible goals (which goals make sense), expectancy (what will happen next), and typical courses of action COA (what actions worked before). In the evaluation phase, a decision maker evaluates the recognized courses of actions one by one until a workable solu-

tion is obtained. Due to the dynamic and uncertain nature of the environment, a decision maker keeps monitoring the status of expectancy so that the situation can be further diagnosed in case that the decision maker had misinterpreted the situation. Similar to case-based reasoning, the RPD model stresses on Simon’s satisficing criterion [13] rather than optimizing in option evaluation.

The RPD framework has been widely employed to develop cognitively inspired software agents [10, 16, 14, 9, 2, 3]. Warwick et al. [16] proposed a decision-specific RPD architecture, encoding experts’ experience as traces in a long term memory (LTM) structure based on Hintzman’s multiple trace memory model [5]. Sokolowski [14] employed a composite agent approach, where the RPD decision process was captured by five embedded agents. Fan et al. [2, 3] have implemented R-CAST to support human-agent decision-making teams, arguing that managing the RPD decision process at the team level enables team members (humans and agents) to establish a shared mental model about the dynamic progress of joint activities. However, all these attempts lack either teamwork support (distributed cognition) or timeliness support, and none of them explicitly employ RPD as the kernel of their architecture.

### 3. Collaboration Net and RPD Process

To capture timing constraints of a process, we choose to extend the notion of timed transition Petri nets (TTPT) [11]. A TTPT is a tuple  $\langle P, T, A, f_0, \gamma \rangle$ , where  $P$  is a finite set of token places;  $T$  is a finite set of transitions;  $A \subseteq (P \times T) \cup (T \times P)$  is a set of directed arcs connecting places and transitions;  $f_0$  is the initial marking (specifying the number of tokens in each place  $p \in P$ ); and  $\gamma$  is a mapping that associates each  $t \in T$  with a time duration. The input places (pre-places) of a transition  $t$ , denoted by  $\bullet t$ , are the places from which an arc runs to  $t$ ; the output places (post-places) of  $t$ , denoted by  $t \bullet$ , are the places to which arcs run from  $t$ . ‘ $\bullet$ ’ is also used for pre-(post-) transitions. We extend the timed transition Petri net with some new concepts.

**Definition 1 (Token Types)** A local token  $\theta$  is a token that flows only within the net where it is originated; A sync token  $\pi$  is a token received from another peer agent through a synchronization message; A team token  $\omega_G$  with respect to a group  $G$  of agents is a meta token which contains exactly one token slot for each agent  $a \in G$ .

Let  $\Theta$ ,  $\Pi$ , and  $\Omega$  be the universe of local tokens, sync tokens, and team tokens, respectively. Let  $\omega_G(a)$  return a reference to the token in the slot for agent  $a$  in  $\omega_G$ .  $\omega_G(a) = \text{null}$  if the slot for  $a$  has no token.

**Definition 2 (Fulfillment)** A team token  $\omega_G$  is fulfilled iff (a)  $\forall a \in G \cdot \omega_G(a) \neq \text{null}$ , (b)  $\exists! b \in G \cdot \omega_G(b) \in \Theta$ , and (c)  $\forall a \neq b \in G \cdot \omega_G(a) \in \Pi$ .

That is, a team token is fulfilled iff there is exactly one token from the local agent and one sync token from each of the group members through synchronization messages.

**Definition 3 (Node Types)** A local node is a Petri net place that takes local tokens only. A team node with respect to a group of participating agents is a Petri net place that takes local tokens (when a transition is fired) and sync tokens (when a message is received) to produce a team token and manages its fulfillment.

A transition can trigger the execution of an activity. An activity can be performed by agents individually, or jointly performed by a group of agents. Each activity  $\alpha$  is associated with a role constraint  $\rho(\alpha)$ , which specifies the capability requirements on an agent. We use  $\text{CanDo}(a, \alpha)$  to indicate that agent  $a$  satisfies  $\rho(\alpha)$  (i.e., can do  $\alpha$ ). We also assume that each activity  $\alpha$  is guarded with a collection of first-order expressions, which is denoted by  $\text{pre}(\alpha)$ . An agent  $a$  cannot perform  $\alpha$  if  $\text{pre}(\alpha)$  is false when it is evaluated relative to  $a$ ’s mental state.

Because a group of agents need to synchronize their activities, we assume there is a global timing mechanism in the system and define timers in terms of global time points.

**Definition 4 (Timer)** A timer  $\xi$  is a tuple  $\langle s, e \rangle$ , where  $s$  and  $e$  are the starting and ending global time points, respectively.

Let  $s(\xi)$  and  $e(\xi)$  return the starting and ending points of  $\xi$ , respectively, and  $\Xi$  be the universe of timers.

**Definition 5 (Collaboration Net)** A collaboration net is an extended TTPT  $\langle L, M, T, A, f_0, \gamma, \delta \rangle$ , where

- $L$  is a finite set of local nodes;
- $M$  is a finite set of team nodes. For each  $m \in M$ , let  $G(m)$  returns the group of participating agents (which can be dynamically assigned);
- $T$  is a finite set of transitions, each of which is associated with an activity. For each transition  $t \in T$ , let  $\alpha(t)$  return the associated activity;
- $A \subseteq ((L \cup M) \times T) \cup (T \times (L \cup M))$  is a set of directed arcs connecting places and transitions;
- $f_0$  is the initial marking;
- $\gamma : T \rightarrow \Xi$ .  $\gamma(t)$  returns the timer associated with transition  $t$ ; and
- $\delta : M \rightarrow \Xi$ , where  $\delta(p)$  returns the timer associated with team node  $p$ .

We say that a group of agents share a collaboration net iff each agent has a local copy of the net and they all agree on the global timing constraints as given by  $\gamma$  and  $\delta$ .

**Definition 6 (Transition Enableness)** A transition  $t$  of a collaboration net is  $R$ -enabled w.r.t. agent  $a$  iff (i)  $CanDo(a, \alpha(t))$  holds; (ii)  $pre(\alpha(t))$  is evaluated by  $a$  as true; (iii) for any  $p \in \bullet t$ , there is a token in  $p$ ; and (iv) for any  $p \in \bullet t \cap M$ , the team token in  $p$  is fulfilled.

**Definition 7 (Transition Semantics)** When firing a transition  $t$ , an agent  $a$  will (i) immediately remove the tokens from its input places  $\bullet t$ ; (ii) idle if  $s(\gamma(t))$  has not come; (iii) when  $s(\gamma(t))$  comes, start executing the associated activity  $\alpha(t)$ ; (iv) once  $e(\gamma(t))$  comes, stop the activity, deposit one token in each of the output places  $t\bullet$ , and for any  $p \in M \cap t\bullet$ , the slot of the team token for the local agent  $a$  is filled, and the agent sends a synchronization message with a sync token to each agent peer in  $G(p)$  excluding itself; and (v) if the current time falls beyond  $\gamma(t)$ , skip the activity, deposit one token in each of the output places  $t\bullet$ , and for any  $p \in M \cap t\bullet$ , fill the slot of the team token for the local agent, and send a synchronization message with a sync token to each agent peer in  $G(p)$  excluding itself.

**Definition 8 (Time-adaptable Team Net)** A time-adaptable team net is a collaboration net where a transition fires as soon as it is  $R$ -enabled.

A time-adaptable team net allows an agent to conduct minimal chores (e.g. synchronization, belief update) before proceeding forward. Since all the activities with broken timers are skipped, a delayed agent could catch up with the other peer agents while still honoring the operation sequence. A time-adaptable team net offers a compromised solution when both timeliness and team coherence are the keys to teamwork success.

### 3.1. Team RPD Process

In order to support the orchestration of a decision making group (e.g., involving agents and human), we develop a timed Petri-net based RPD process and adapt it to teamwork settings.

**Definition 9 (Team RPD Process)** A Team RPD process is a time-adaptable team net  $\varpi = \langle L, M, T, A, f_0, \gamma, \delta \rangle$  as depicted in Figure 1, where

- $T = \{t_i | 0 \leq i \leq 11\}$ , where for  $i$  from 0 to 11,  $\alpha(t_i)$  refers to SituationExperiencing, FeatureMatching, StoryBuilding, RecognitionConsolidating, ExpectancyMonitoring, COASelecting, COAAdjusting, COAEvaluating, RecognitionChecking, RecognitionRefining, SituationRefreshing, and COAExecuting, respectively.  $\alpha(t_3)$ ,  $\alpha(t_7)$ , and  $\alpha(t_8)$  are team activities;
- $L = \{p_0, p_1, p_7, p_8\}$ ;

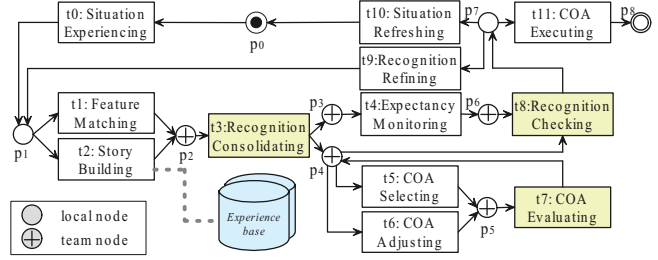


Figure 1. The RPD Cognition Process

- $M = \{p_2, p_3, p_4, p_5, p_6\}$ , where  $G(p_2)$  is the group of agents taking the role of performing  $t_3$ ,  $G(p_3)$  is the group of agents taking the role of performing  $t_4$ ,  $G(p_4)$  is the group of agents taking the role of performing  $t_5$  and  $t_6$ ,  $G(p_5)$  is the group of agents taking the role of performing  $t_7$ , and  $G(p_4) \cup G(p_6)$  is the group of agents taking the role of performing  $t_8$ ;
- $A$  and  $f_0$  are obvious,  $\gamma$  and  $\delta$  are problem specific.

Given a decision task  $d$ , we assume the presence of an experience base  $\Delta_d$ —a collection of decision making experience—as pertinent to  $d$ . The nature of a decision task determines the collection of features (predicates) required to fully describe and understand a situation. Let  $\Psi(d)$  be such a collection for task  $d$ .  $\Psi(d)$  can have a structure among the features (e.g. layered structure reflecting information production chains). Each experience is a tuple  $\langle S, Act \rangle$ , where  $S$  is a collection of instantiated predicates, describing a specific situation, and  $Act$  is a set of course of actions (COA) that worked before under situation  $S$ . Let  $\eta_d$  be a subjective threshold such that a reasonable explanation can be drawn from a situation description  $S$  when  $|S| \geq \eta_d$ . Typically,  $\eta_d \leq |S| \leq |\Psi(d)|$ .

The activity SituationExperiencing is performed by an agent to collect and synthesize (situation awareness) relevant information to produce a description of the current situation  $S_0$ . When  $\eta_d \leq |S_0|$ , FeatureMatching is used by an agent to match  $S_0$  with past experience in order to identify similar cases. When  $\eta_d > |S_0|$ , StoryBuilding is used to develop a potential explanation of  $S_0$ .

RecognitionConsolidating is a team activity, which is responsible for generating the four recognition products (ref. Sec. 2): a set  $E$  of expectancy, a set  $C$  of relevant cues, a set  $G$  of plausible goals, and a set  $\Lambda$  of COAs (object-level processes) as drawn from the matched experience and customized to the current situation. A team policy can be employed by a group of agents to negotiate on the products.  $\Psi(d) \setminus S_0$  is a collection of missing information in  $S_0$ . Those with a future time point as an argument (what will happen next) can be used to generate  $E$ , while others can be used to generate  $C$  for diagnosing the situation.

ExpectancyMonitoring and COAEvaluating are parallel activities. In ExpectancyMonitoring, an agent keeps monitoring the expectancy  $E$  as the situation changes, broadcasts a violated expectancy (anomaly) to agents in  $G(p_6)$  whenever it emerges, and produces a set  $\chi \subseteq E$  of violated expectancy. COASelecting maintains a selected COA  $\lambda$  and a candidate set  $\Lambda_C$ . Initially  $\Lambda_C = \Lambda$  and  $\lambda = null$ . When COASelecting fires, a new  $\lambda$  is drawn (removed) from  $\Lambda_C$ . In COAAdjusting, an agent adjusts the selected  $\lambda$ .

COAEvaluating is a team activity. It is responsible for resolving uncertainties regarding the selected COA  $\lambda$ . In particular, a group of agents need to collaborate on four activities as related to the potential intention to do  $\lambda$ . First, determining who will assume the responsibility of doing  $\lambda$ . When more than one agents take the role, they have to communicate to establish mutual beliefs regarding each other's commitment to the role. Second,  $\lambda$  is a domain-level process that can have timing constraints. All the involved agents need to check whether the timing constraints can be honored. Third, it has to be checked that the execution of  $\lambda$  will keep the achievability of all the goals in  $G$ . A symbolic reasoning engine can be employed for this purpose. Fourth, each potential doer also needs to make a commitment to informing others whenever it individually figures out that the constraints of doing  $\lambda$  no longer hold. This will allow the team to terminate the evaluation of  $\lambda$  and start to consider another COA at the earliest possible opportunity. In the end, if  $\lambda$  is accepted as a solution,  $Workable(\lambda)$  is asserted; if  $\lambda$  is not a solution, it can be adjusted and reevaluated in the next iteration. Setting  $\lambda$  to null will trigger the selection of a new COA.

RecognitionChecking is a team activity and it is a point to synchronize the monitoring and evaluation processes. It is responsible for checking the achievability of plausible goals in  $G$ .  $G$  has been checked in  $t_7$ . However, it is necessary to check it again because (a) the group of agents reaching  $t_8$  can be different (e.g., a super set) from those reaching  $t_7$ , and (b) the situation might have changed. Again, a team policy can be employed to do so. In the end, RecognitionChecking establishes a group commitment to  $\lambda$  (with  $Committed(\lambda)$  asserted) if  $G$  is achievable and no anomaly occurs so far.

COAExecuting is performed to implement the committed COA, which can involve multiple agents (and the human counterparts). RecognitionRefining is performed to further diagnose the situation. SituationRefreshing is performed to start a new round of situation recognition.

In Figure 1,  $\{t_1, t_2\}$ ,  $\{t_5, t_6, t_8\}$ , and  $\{t_9, t_{10}, t_{11}\}$  are collections of transitions sharing input places. Typically, a conditional structure can be captured by multiple transitions sharing the same set of input places. However, an agent can get blocked due to lack of condition if the associated preconditions are not well structured.

**Definition 10** A well-structured net is a net such that for any collection  $U$  of transitions, if they share the same set of input places, their preconditions complement. Formally, if  $\forall U \subseteq T, \exists P \subseteq L \cup M$  such that (i)  $(\forall t \in U \cdot \bullet t = P)$  and (ii)  $(\forall p \in P \cdot p \bullet = U)$ , and  $U$  and  $P$  are the smallest sets satisfying (i) and (ii), then  $\bigvee_{t \in U} pre(\alpha(t))$  is always true.

**Property 1** The net in Figure 1 is well-structured with the assignments below:

$$\begin{aligned} pre(FeatureMatching) &\triangleq \eta_d \leq |S_0|; \\ pre(StoryBuilding) &\triangleq \eta_d > |S_0|; \\ pre(COASelecting) &\triangleq (\Lambda_C \neq \emptyset) \wedge (\lambda = null); \\ pre(COAAdjusting) &\triangleq (\lambda \neq null) \wedge \neg Workable(\lambda); \\ pre(RecognitionChecking) &\triangleq \\ &((\Lambda_C = \emptyset) \wedge (\lambda = null)) \vee \\ &((\lambda \neq null) \wedge Workable(\lambda)) \vee (\chi \neq \emptyset); \\ pre(COAExecuting) &\triangleq \\ &(\chi = \emptyset) \wedge \exists \beta \in \Lambda \cdot Committed(\beta); \\ pre(RecognitionRefining) &\triangleq (\chi \neq \emptyset) \wedge \\ &\exists g \in G \cdot Unachievable(g); \\ pre(SituationRefreshing) &\triangleq \exists \beta \in \Lambda \cdot Committed(\beta) \vee \\ &((\chi \neq \emptyset) \wedge \exists g \in G \cdot Unachievable(g)). \end{aligned}$$

Bellow, we assume  $\varpi$  is a well-structured team RPD net.

**Definition 11** A multi-agent system collaborating through  $\varpi$  is dead iff there exists one agent being blocked at a team node of  $\varpi$ .

**Property 2** Given that a group of agents  $\{a_1, a_2, \dots, a_n\}$  collaborate through  $\varpi$ . Suppose in  $\varpi$ , node  $p$  starts a parallel branch which contains a team node  $m$ , and let  $T' = \{t \in T | p \in \bullet t\}$ . The system is dead if there exists an agent  $a_i$  such that (i)  $a_i \in G(m)$ , and (ii)  $\forall t \in T' \cdot \neg CanDo(a_i, \alpha(t))$ .

*Proof Sketch:* Suppose  $a_k \in G(m)$  and  $a_k$  has entered the parallel branch (in its own copy of  $\varpi$ ) and is waiting at  $m$  for a sync token from agent  $a_i$ . However, for  $a_i$ , no transition in the parallel branch can be enabled due to lack of capability (Condition (ii)), thus  $a_i$  cannot send out a sync token to  $a_k$ , who will be blocked forever.  $\square$

For instance, in Fig. 1, if agent  $a \in G(p_5)$  but cannot do COASelecting and COAAdjusting,  $a$  could make other agents blocked at  $p_5$ .

**Property 3** Assume that an agent cannot get blocked due to lack of condition or capability. A multi-agent system collaborating through  $\varpi$  can honor  $\varpi$ 's timing constraints.

*Proof Sketch:* According to the firing semantics of transitions and the assumption, the only possibility of breaking a timing constraint is at team nodes. We also assumed that the communication is reliable. When there is no delay of sync tokens, the team token at a team node can be

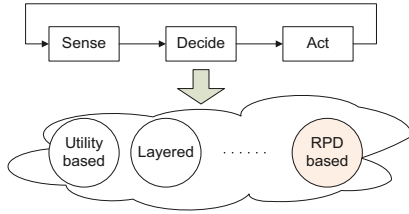


Figure 2. Agent Architecture

fulfilled immediately, then all the timing constraints can be fully honored. When there is communication delay, suppose some sync token for node  $m$  is the last one arrived at an agent  $a$  and  $e(\delta(m))$  is already passed. In such a case, it is possible that agents other than  $a$  are still on schedule. For those agents like  $a$  who are lagging behind will speed up while still honoring the firing sequence, they will eventually catch up with other agents and honor the rest of the timing constraints. □

It is highly likely that lagging happens randomly to the group of agents. In such a case, the system can still offer services with a desirable level of quality when a role is played by multiple agents.

#### 4. Recognition-Primed Agent Architecture

An agent architecture typically takes the standard sense/decide/act loop as shown in Figure 2, where the “decide” module can be replaced by concrete inference mechanisms (e.g., game theory or POMDP for utility-based agent architecture, intention generation for BDI architecture, reflection-deliberation for layered architecture).

We argue that the RPD model can be employed as a generic architecture for cognitive agents for the following reasons. First, human beings are not ideal decision makers. When making decisions, people usually do not know all the possible choices or their probabilities, and they rarely use classical decision theories to arrive at a decision [6, 8]. Contrastly, cognitively inspired models like RPD take a more human-like approach where the decision making processes of an agent and its human peer can be easily meshed with each other, leading to *human-aware computing* with improved trustworthiness and system performance.

Second, the RPD model offers a well-structured *macrocognitive process* that captures the cognitive activities undergoing in the mind of an individual decision maker. Computational RPD models can be constructed from this macrocognitive process such that human and agent could adapt their work progress to the changing time stress and collaborate under the same context. In addition, the RPD process can be extended into a team process where multiple agents (including both agents and human) could

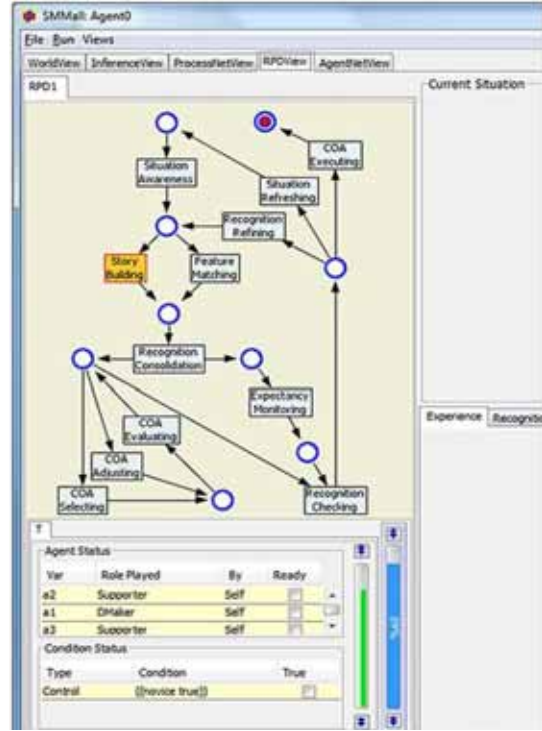


Figure 3. Team RPD Process in SMMall

synchronize their teamwork behavior and exchange experience/knowledge/information along the decision process. Consequently, a multi-agent system build upon the team RPD process is able to progressively establish and maintain a shared mental model about the ongoing activities.

Third, the timeliness and quality of decision making is critical to teams in time-stressed domains such as disaster response, military command and control. The RPD model exactly focuses on how people actually make decisions in realistic settings that typically involve ill-structured problems, uncertain dynamic environments, shifting/competing goals, and time stress. A RPD-enabled recognition-primed architecture would enforce distributed agents to honor the time-stress feature of realistic decision making tasks. As cognitive aids, it is also feasible for an agent to learn realistic cognitive models of its human peer (e.g. model of cognitive inclination). Such models would allow an agent to offer nonintrusive help when its human peer is under time stress or cognitively overloaded.

We have implemented SMMall (Shared Mental Models for all) – a system for developing cognitive agents. Figure 3 gives a screenshot of SMMall agent, the kernel of which implements the team RPD process as formalized in Sec. 3.

Process modeling in SMMall involves a 2-dimensional process space: abstraction dimension and composition dimension. Along the composition dimension, a (complex) process can be decomposed into sub-processes at the same

abstraction level. Along the abstraction dimension, we distinguish processes at the meta level, cognition level, and domain level. The team RPD process is a meta-level process, which serves as a mechanism for coordinating lower-level processes and offers a common ground for all the agents/human to anchor their recognition context. Cognition-level processes refer to those activities (courses of action) associated with each transition of the RPD process. The procedural knowledge captured by a cognition-level process can be applied in various domains. For instance, FeatureMatching may involve feature selection, information-need analysis, and experience filtering. Note that different agents may have different cognition-level processes, and a cognition-level process (e.g., COAEvaluating, COAAdjusting) may involve the manipulation of domain-level processes (e.g, removing threats in the field).

All processes in SMMall are modeled as time-adaptable team net, where each place and transition is associated with a human-adjustable timer. Through the interface, a human user can easily navigate processes at different levels and is able to collaborate with an agent on cognitive activities such as situation evaluation, COA adjusting, experience filtering, and recognition refining. A person may be involved in multiple decision making groups and playing different roles at the same time. SMMall supports parallel RPD processes and allows a human user to switch among multiple on-going decision making tasks.

## 5. Conclusion

Developing multi-agent systems for human-centric teamwork is extremely challenging. It mandates the integral consideration of architectural flexibility, teamwork adaptability, and the self-management of collaboration contexts.

We extended the concept of timed Petri nets to explicitly capture timing constraints and collaboration points along the RPD process. The team RPD process can serve as a *paradigm* for designing multi-agent systems that support collaborative decision making activities in dynamic, time-stressed domains.

We also argued that the RPD model can serve as the kernel of cognitive agent architectures, which allows a group of agents and their human peers to interact with each other and adapt to collaboration needs from other members. The framework not only lays a foundation for building human-centric multi-agent systems, it also offers a practical coordination and cooperation mechanism for agent systems that need to operate in uncertain, time-stressed domains.

## References

[1] J. R. Anderson and C. Lebiere. *The atomic components of thought*. Mahwah, NJ: Erlbaum, 1998.

[2] X. Fan, S. Sun, M. McNeese, and J. Yen. Extending the recognition-primed decision model to support human-agent collaboration. In *AAMAS'05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 945–952. ACM Press, 2005.

[3] X. Fan and J. Yen. R-CAST: Integrating team intelligence for human-centered teamwork. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI'07)*, pages 1535–1541, 2007.

[4] B. J. Grosz, S. Kraus, S. Talman, B. Stossel, and M. Havlin. The influence of social dependencies on decision-making: Initial investigations with a new game. In *Proceedings of the third international joint conference on Autonomous Agents and Multiagent Systems*, pages 782–789, 2004.

[5] D. L. Hintzman. Judgments of frequency and recognition memory in a multiple-trace memory model. *Psychological Review*, 95:528–551, 1988.

[6] G. A. Klein. Recognition-primed decisions. In W. B. Rouse, editor, *Advances in man-machine systems research*, volume 5, pages 47–92. Greenwich, CT: JAI Press, 1989.

[7] J. Laird, A. Newell, and P. Rosenbloom. Soar: an architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[8] B. Mellers, A. Schwartz, and A. Cooke. Judgement and decision making. *Annual Review of Psychology*, 49:447–478, 1998.

[9] E. Norling. Folk psychology for human modelling: Extending the BDI paradigm. In *AAMAS '04: International Conference on Autonomous Agents and Multi Agent Systems*, pages 202–209, 2004.

[10] E. Norling, L. Sonenberg, and R. Ronnquist. Enhancing multi-agent based simulation with human-like decision making strategies. In S. Moss and P. Davidsson, editors, *Proceedings of the Second International Workshop on Multi-Agent Based Simulation*, pages 214–228, 2000.

[11] C. Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. Technical Report MAC-TR-120, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.

[12] F. E. Ritter, editor. *Techniques for Modeling Human Performance in Synthetic Environments: A Supplementary Review*. 2002.

[13] H. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[14] J. Sokolowski. Enhanced military decision modeling using a multiagent system approach. In *BRIMS'03: Proceedings of the Twelfth Conference on Behavior Representation in Modeling and Simulation*, pages 179–186, 2003.

[15] M. Tambe. Towards flexible teamwork. *Journal of AI Research*, 7:83–124, 1997.

[16] W. Warwick, S. McIlwaine, R. Hutton, and P. McDermott. Developing computational models of recognition-primed decision making. In *Proceedings of the tenth conference on Computer Generated Forces*, 2001.

[17] J. Yen, J. Yin, T. Ioerger, M. Miller, D. Xu, and R. Volz. Cast: Collaborative agents for simulating teamworks. In *Proceedings of IJCAI'2001*, pages 1135–1142, 2001.

# Using Knowledge Objects to Exchange Knowledge in a MAS platform

Ana Paula Lemke and Marcelo Blois

PPGCC – PUCRS, Av. Ipiranga 6681, Partenon, 90.619-900, RS, Brazil

{ana.lemke, marcelo.blois}@pucrs.br

## Abstract

*Agent-based platforms and toolkits have been proposed to help developers to create agent-based applications, but there is a visible separation between theory and practice. Some challenges, such as representing the knowledge used by software agents in a structure suitable for it to be reused and shared, still need to be addressed by the agent platforms. This paper describes the main features of the SemantiCore+Ontowledge, an agent infrastructure to develop multi-agent systems in which the agents can manage their knowledge and perform in a Semantic Web context. It is proposed the use of knowledge objects to encapsulate and organize the knowledge available using an ontology-based knowledge representation.*

## 1. Introduction

The Semantic Web brings structure to the meaningful content of Web pages and creates an environment where software agents can readily carry out sophisticated tasks for users [1]. In fact, many descriptions of the Semantic Web include the use of agents as an enabling technology for delivering services to the users [2]. The most common way to enable agents to interoperate over the Semantic Web is to give each of them the same specified conceptualization of the domains they are expected to work in [3]. This ontology can represent public information or an agreed set of definitions and meanings of basic communicable concepts. Thus it would be desirable for agent infrastructures to support the development of knowledge-aware systems so that agents can collect Web content (shared knowledge) from diverse sources, process the information and exchange the results with other agents.

Other challenging problem in the deployment of open and flexible systems using the Web infrastructure is the development of agent platforms that provide mechanisms for the agents to manage their knowledge. A promising way to solve the problem of improving the agents' behavior in a MAS is explicitly representing the domain knowledge and making this knowledge available to other agents. So, besides the basic requirements every agent platform must have such as agent communication and coordination, we believe that the development of MAS also include knowledge management (KM) activities to specify the

knowledge the agents need to reason in order to interoperate and act.

This paper introduces the SemantiCore+Ontowledge framework which allows the development of software agents capable of self-manage their knowledge. A SemantiCore+Ontowledge agent can search and apply updated knowledge at run-time for achieving a certain goal and it is capable of evaluating its execution results to identify when the current knowledge must be replaced for other knowledge. The motivation for implementing these operations in an agent platform was based on human's usual behavior: (i) when we do not know how to do something, we usually look for information in books, at web sites, or ask someone who does it; (ii) if we do not obtain good results doing something, we usually look for another way to do it better. Native knowledge search and exchange is not present in most agent platforms available.

The SemantiCore+Ontowledge is an extensible framework where component organization, different hot spots instantiation, KM functionalities, and native OWL processing contribute to differentiate it from other agent platforms. In our approach, rather than modeling mental attributes of agents, we concentrate on the way knowledge is distributed among agents and how it can be used and learned efficiently.

This paper is structured as follows. Section 2 presents the architectural requirements. Section 3 describes the SemantiCore+Ontowledge framework architecture. Section 4 presents a discussion about the proposed framework and the case studies developed. Section 5 describes some related works and section 5 concludes the paper.

## 2. Architectural requirements

In order to adapt or learn, the agents should be able to obtain knowledge and dynamically change their behavior. The appropriated knowledge could be obtained by cooperating with other agents (knowledge exchange) or by new knowledge creation (possibly based on algorithms for machine learning and inference engines). Focusing on the first alternative, we compiled some requirements an agent platform must have to enable knowledge exchange.

**Requirement 1.** *The agent platform shall provide domain knowledge representation to the agents. The agent's knowledge must be encoded using a formal*

language. It is essential for agents to interpret the knowledge available and infer new knowledge.

**Requirement 2.** *The agent platform shall provide an outline for creating knowledge packages.* The knowledge must be divided into slices or knowledge packages. Each knowledge package must contain all knowledge necessary for an agent to achieve a goal. Some researchers investigate the exchange of action plans among agents [4, 5]. We believe that an action plan represents a fragment of the necessary knowledge<sup>1</sup> for an agent to achieve a goal. A knowledge package should include an action plan, rules, beliefs, sensors, effectors, and so on.

**Requirement 3.** *The agent platform shall offer facilities for agents to exchange knowledge between each other.* Agents are expected neither to possess the capability of looking for external knowledge nor to dynamically extend their knowledge base with the knowledge retrieved from other agents. To exchange knowledge and use the retrieved knowledge, the agents should be able to execute a set of tasks. Several questions need to be addressed in engineering MAS where knowledge exchanges can happen. For instance, it must be decided which knowledge can be exchanged, and how to understand and use the knowledge retrieved from another agent.

In the SemantiCore+Ontowledge framework, the agent's knowledge is represented as ontologies (requirement #1), it is defined the notion of knowledge objects (requirement #2), and there are methods for allowing knowledge objects sharing and use (requirement #3). The next section will explain how SemantiCore+Ontowledge framework implements the cited requirements.

### 3. SemantiCore+Ontowledge Framework

The SemantiCore+Ontowledge (or simply S+O) is a Java framework designed to allow agents to share and use knowledge in MASs. It provides a better support for agents that do not possess the necessary knowledge for achieving a certain goal. In the literature, there is a lack of proposals to integrate a KM process in a MAS architecture allowing agents to encapsulate the knowledge available in a format suitable for it to be stored and exchanged among other agents. The creation of agent platforms specifically designed for supporting the deployment of agent systems in the Semantic Web is also a subject that needs more attention from the agent community.

The proposed framework allows the creation of agents that use a defined KM process to make explicit and share knowledge. KM can be defined as “the identification and analysis of available and required knowledge assets and knowledge asset related processes, and the subsequent planning and control of actions to develop both the assets and the processes so as to fulfill organizational objectives”

---

<sup>1</sup> Our notion of knowledge is similar to the one used in [6] for information in action, i.e. information applied to a specific purpose.

[7]. This definition of KM implies that it is necessary for business organizations to: (i) be capable of identifying and representing their knowledge assets; (ii) share and reuse these knowledge assets for differing reasons and by different users; and (iii) create a culture that encourages knowledge sharing and reuse [8].

Mapping a KM process in a software architecture could help agents to share knowledge. However, the KM process must be mapped to the characteristics of MAS organizations. For instance, there is no meaning to talk about implementing a culture for encouraging knowledge sharing and reuse in MAS organizations. A MAS organization needs to be capable of representing and sharing its knowledge assets.

#### 3.1. The S+O agent components

S+O agents are formed by five basic components. The *Sensorial* component is responsible for sensing the environment. It manages all the different sensors an agent has, selecting one of them according to the kind of message received from the environment. The *Decision* component handles the rules and facts that form the mental model of an S+O agent. The Decision controls the agent goal's execution. The S+O agents have goals (represented as an ontology instance) and these goals can be achieved by the execution of an action plan.

The *Execution* component manages the execution of the agent's actions allowing developers to access all the agent's primitives. Actions may need to send messages to other agents as a part of their processing. The encapsulation of these messages in different structures and their transmission in the environment is the responsibility of the *Effector* component. When an agent is created, the developer may indicate the types of effectors the agent will work with. This feature allows an agent to talk simultaneously with different peers such as Web services (SOAP messages) and agents created in a FIPA-compliant platform (ACL messages).

The *Organizer* component includes mechanisms for allowing agents to retrieve external knowledge to achieve their goals. The agent's knowledge is represented as *knowledge objects* that are stored in the agent's knowledge base (*internal knowledge base*). The KOs can be retrieved from other agents or from the *environment knowledge base*. *Execution records* are used to proactively start the searching for “better” KOs. Knowledge acquisition and loading are enabled by the *knowledge acquisition and loading mechanisms*. The agents can share KOs using the *knowledge sharing mechanism*. In the next sections, we provide a detailed discussion of these concepts.

#### 3.2. Knowledge Object Structure

The KOs encapsulate the basic elements to achieve a certain goal (solve a problem). They are represented using an ontology-based knowledge representation. Ontologies



are a promising technology for information sharing and reuse [9]. Basically, each KOs can be divided into four main parts: (i) a problem description; (ii) a list of meta-data; (iii) an access modifier; and (iv) a solution for the problem described.

The first three parts are represented as *Goal\_Item*, *Description\_Item* and *Restriction\_Item*. The solution description is formed by all other architecture-dependent elements in the KO (these elements are loaded in the agent's architecture when a specific KO is applied). For instance, in a BDI agent, the architecture-dependent KO items would be beliefs, desires and intentions.

The *Goal\_Item* is used in association with the *KO goal*. The KO goal (like the agent goal) is represented as an ontology instance that describes the concepts involved in the problem. The KO goal is compared to the ontology which describes the problem trying to be solved to check if the KO is suitable to solve it. The *Goal\_Item* also contains some criteria to evaluate the KO execution. These criteria are used to check if the goal was satisfactory achieved or not. To make a KO evaluation, an agent: (i) uses the knowledge encapsulated into a KO to achieve a specific goal; (ii) recovers its criteria for evaluating goals achievement; and (iii) compares the information derived from goal's execution to compute the KO execution grade.

The *Description\_Item* represents meta-data about the KO. These meta-data were selected among the set defined by the Dublin Core Metadata standard ([www.dublincore.org](http://www.dublincore.org)). An agent can use these data as additional information to select the most appropriate KO to use or share. The *Restriction\_Item* represents KO sharing restrictions. It indicates the set of agents the KO can be shared with. Sharing restrictions are verified by the knowledge sharing mechanism when an agent wants to share a KO with other agent in the organization.

The KOs can contain architecture-dependent items of five types (sensors, effectors, rules, facts and action plans). Each architecture-dependent item has two attributes: `element`, that represents an agent's architectural constructor; and `elementSchema`, that contains an ontology with all necessary information to recreate the agent's architectural constructor in other agents. During knowledge transmission the `elementSchema` is instantiated with the information provided by the `element` attribute. The instantiated model is the unique information transmitted through the environment.

To implement an architecture-dependent item is necessary to extend the `KOItem` class. The `KOItem` class contains four abstract methods to serialize, parse, load and remove a specified `element`. The `serialize` method enables KO items transmission. The `parse` method indicates how to understand the knowledge retrieved from another agent. The `load` method describes how to use the knowledge retrieved (how to load it in the agent's architecture). The `remove` method is responsible for eliminating the item of the agent's architecture.

If an agent wants to sell books, for example, it will need - among other items - a sensor to capture messages whose subject includes the text "*book requisition*". Then the *Books\_Selling* KO - which encapsulates the knowledge related to books selling - should have a "*RequisitionSensor*" sensor to capture *string* messages whose subject contains the *string* literal "*book requisition*".

To share a KO it is necessary to serialize all its items. Then, for an agent to share the *Books\_Selling* KO it must serialize its "*RequisitionSensor*" sensor. To serialize a *Sensor\_Item* is necessary to gather: the sensor's name; the sensor's class (sensors are hot spots in the S+O framework and they are created extending the `sensor` class); and the sensor's header and content patterns (these patterns are used to filter the messages received from the environment). The Figure 1 shows a fragment of the OWL code that was generated for sharing the *Books\_Selling* KO.

When the *Books\_Selling* KO was parsed in the receiver agent (after transmission), a new `StringSensor` object will be created with the name "*RequisitionSensor*" and the header pattern ("*?message*", "*subject*", "*book requisition*"). If the *Books\_Selling* KO was selected to be applied, then the "*RequisitionSensor*" sensor will be added in the agent's sensors list. Thus the agent will be able to capture messages whose content contains the *string* literal cited.

### 3.3. Knowledge bases

The internal knowledge base represents the agent's knowledge base. It is dynamically extended with the KOs retrieved from other agents. Though it is possible to implement a procedure for discarding a KO after its usage. This can be useful in computational devices with limited physical resources such as *Personal Digital Assistants* (PDAs).

The *Librarian* agent represents the environment knowledge base and it is an administrative agent in S+O framework. The Librarian's knowledge base contains only approved KOs. An approved KO was evaluated and confirmed by a domain expert. The knowledge retrieved from the *Librarian* agent may be considered more "trustworthy" since it was evaluated by an expert. On the other hand, other agents can provide more updated KOs. We will implement interfaces for KOs uploading, creating and editing in the near future.

### 3.4. Execution records management

Each time a KO is executed an *execution record* is created with the results of the execution. The execution records allow an agent to create policies for requesting new KOs to achieve a particular goal based on previous execution results. The *changing policy* defines how the execution results are evaluated. For instance, if the *Books\_Selling* KO execution was evaluated in a numeric scale from 1 to 5, then a possible changing policy would be

```

<rdf:Description rdf:about="http://semanticore.pucrs.br#KO3">
  <j.0:hasItem rdf:resource="http://semanticore.pucrs.br#KO3_item2"/>
  <j.0:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Books_Selling</j.0:name>
  ...
  <rdf:type rdf:resource="http://semanticore.pucrs.br#KnowledgeObject"/>
</rdf:Description>

<rdf:Description rdf:about="http://semanticore.pucrs.br#KO3_item2">
  <j.0:element rdf:resource="http://semanticore.pucrs.br#RequisitionSensor"/>
  <j.0:itemClass rdf:datatype="http://www.w3.org/2001/XMLSchema#string">agent.organizer.hotspots.SensorITEM</j.0:itemClass>
  <rdf:type rdf:resource="http://semanticore.pucrs.br#KOItem"/>
</rdf:Description>

<rdf:Description rdf:about="http://semanticore.pucrs.br#RequisitionSensor">
  <j.0:headerPattern rdf:resource="http://semanticore.pucrs.br#fact21"/>
  <j.0:sensorClass rdf:datatype="http://www.w3.org/2001/XMLSchema#string">sensorial.hotspots.StringSensor</j.0:sensorClass>
  <rdf:type rdf:resource="http://semanticore.pucrs.br#Sensor"/>
</rdf:Description>

<rdf:Description rdf:about="http://semanticore.pucrs.br#fact21">
  <j.0:subject rdf:datatype="http://www.w3.org/2001/XMLSchema#string">?message</j.0:subject>
  <j.0:predicate rdf:datatype="http://www.w3.org/2001/XMLSchema#string">subject</j.0:predicate>
  <j.0:object rdf:datatype="http://www.w3.org/2001/XMLSchema#string">"book requisition"</j.0:object>
  <rdf:type rdf:resource="http://semanticore.pucrs.br#SimpleFact"/>
</rdf:Description>

```

Figure 1. Piece of OWL code generated for *Books\_Selling* KO sharing.

to request new KOs to achieve the “*Books\_Selling*” goal after executing the *Books\_Selling* KO with an evaluation equal or less than 3 for 5 times. Using the execution history the agents are able to create more effective recommendations and evolve their behaviors.

### 3.5. Knowledge acquisition and loading mechanisms

The knowledge acquisition mechanism is started when an agent needs: (i) to replace obsolete or unsuitable knowledge (based on the execution history); or (ii) to find knowledge applicable to achieve a goal without knowledge associated. The activities executed for retrieving relevant KOs are described below.

1. The achievement of the goal without prior knowledge is suspended (goals with obsolete knowledge are not suspended while the knowledge acquisition mechanism is executed).
2. The set of addressees for the knowledge request message is defined. The agents have a list of possible addressees for knowledge request messages. This list can have the following values: (1) the identifiers of particular agents in the system; (2) “*all*” (default value), which means that the request message will be sent to all agents found in the yellow pages service (agents whose expertise are related to the agent’s needs); or (3) “*Librarian*”, which is the environment knowledge repository. The resultant addressees list indicates who to ask for the knowledge required.
3. A knowledge request message is created and it is sent to all the agents in the addressees list.

When the requester agent receives a response to a knowledge request, it verifies if the request is not expired and if so it stores the received knowledge. Each agent has an attribute named *waitingTime* that determines the time

spent between the knowledge request message sending and the verification of the received knowledge. The loading mechanism is started when the waiting time ends. The activities executed for selecting and loading the most suitable KO are described below.

1. The agent uses *criteria* for checking the similarity between a requested knowledge and the KOs received. In the S+O framework, both the agent goal and the KO goal are represented as an ontology instance so the criteria must estimate the similarity between two ontologies. An example of criterion used to check the similarity between two ontologies (provided by OntoMetric) is to verify the percentage of common concepts between them [10].
2. The most suitable KO is selected according to a *loading policy*. An example of loading policy could be: “*select the higher graded KO independently of its relevancy value*”. The loading policy must be implemented when the S+O framework is instantiated.
3. The selected KO is loaded in the agent’s architecture. The *load* method (of the *KOItem* class) is used to load all KO items in the agent’s architecture. If one KO item cannot be loaded, other KO must be selected.
4. The selected KO is stored in the agent’s internal knowledge base and it is associated with the goal that started the knowledge acquisition mechanism. If the goal achievement was not suspended, then it is necessary to wait for executing this activity until to accomplish the goal.
5. The goal achievement is restarted.

### 3.6. Knowledge sharing mechanism

When an agent requests knowledge other agents can respond with KOs that are similar to the requested knowledge. The agents have a set of *criteria* to check the similarity between the KOs available and a specific

requested knowledge. The KO sharing restrictions must be verified when an agent wants to share a KO. For instance, some “strategic” KOs can be *private*, which means that they cannot be provided to other agents. To select one or more KOs to share, an agent has a *delivery policy*. An example of delivery policy could be: “*send only the KOs that precisely match the requested knowledge*”.

#### 4. Discussion: S+O Agents in Action

Agents usually cannot deal with the lack of the knowledge required for achieving their goals. The agent platforms typically focus on the distribution issues such as concurrency control, message passing, environment management and internal agent architecture implementation. They do not include mechanisms for managing the agent’s internal knowledge which could allow agents to work together by exchanging knowledge.

Knowledge organization and exchange are important for several reasons. Firstly, the agent’s knowledge can be insufficient to achieve a specific goal. Secondly, the resources required to execute a plan may be unavailable. Finally, the environment is unpredictable and uncertain, which means that it is impossible to predict the future. All of these would possibly result in a situation where the agent’s goal achievement is unfeasible with the knowledge available. Hence, the agent must be able to change its know-how to achieve a certain goal.

The S+O framework facilitates agents’ knowledge organization allowing them to recover and share knowledge at run-time. Since it is a flexible architecture, it has some application specific hot spots. For instance, the criteria for checking the similarity between a requested knowledge and the KOs available are defined as hot spots in S+O framework and thus the developers can implement different algorithms to evaluate the similarity between two ontologies.

It is difficult to evaluate quantitatively the contribution of our proposal to the agent’s execution. Firstly because the application results are strongly related with the context or situation in which the agents are inserted to achieve their goals, i.e., there is a set of metrics whose measures are context-dependent. As mentioned above, the environment is unpredictable and uncertain, which also means that it is impossible to generalize the results obtained. However, we have been developing a number of case studies to investigate whether knowledge exchanges improve the agent’s abilities and to explore the framework’s potential.

The first case study developed was a scenario adapted from [1]. The scenario begins when Pete and Lucy (brother and sister) want to schedule physiotherapy sessions for their mother (named as Marie). Pete and Lucy use their personal “S+O agents” to execute the scheduling task. Nine agents were created for implementing this scenario. *LucyAg* was Lucy’s personal assistant. Among its pre-defined goals there was the “*Clinics Selection*” goal used to find the most

appropriate clinic for a prescribed treatment. *PeteAg* was Pete’s personal assistant that also had the “*Clinics Selection*” goal. The other agents represented the Marie’s doctor, the health care company, the health care services ranking, and 4 different clinics.

Neither *LucyAg* nor *PeteAg* had the knowledge required to achieve the “*Clinics Selection*” goal. Different selection policies were implemented to enable *LucyAg* and *PeteAg* to select different KOs for achieving the “*Clinics Selection*” goal. Two KOs were created and made available: the *KO-1*, that provides treatments scheduling knowledge considering the patient insurance plan and location; and the *KO-2*, that contains an alternative knowledge for treatment scheduling allowing clinics to be directly contacted using a yellow pages service to find them.

Using the *KO-2*, *PeteAg* was able to identify a different set of clinics suitable for Marie’s prescribed treatment than *LucyAg* (in our example, *LucyAg* used the *KO-1* for achieving the cited goal). It does not indicate that the results presented by *KO-1* usage are always worse or better than the results obtained by *KO-2* usage. In the original problem description, Lucy feels comfortable with the clinics selected by her agent. But as Pete is not satisfied with the results given, he demands his agent to redo the searching operation. As a clinic indicated by *PeteAg* was selected for the treatment, we can say that “*the KO-2 was more appropriated to achieve the ‘Clinics Selection’ goal in this context*”. This scenario was implemented using Java and Eclipse. The ontologies were created using OWL DL and processed using the Jena 2.4 API.

Other scenario has been developed to explore the execution history usage. In this scenario there are two vendors of cars and some customers. All vendors and customers have personal S+O agents to help them to sell and buy cars. The vendors’ agents shall act in order to increase the vendors’ profits. On the other hand, the clients’ agents shall negotiate with the vendors’ agents in order to decrease the car cost. If a vendor’s agent does not obtain good results for sometimes (if it does not sell a car to a possible customer or sells it for an unsatisfying price), then it needs to replace its selling strategy to maximize the vendor’s profits. The agent can improve its abilities by using a different KO but it is also possible that it has unsatisfactory results again. However, the S+O agent will evaluate its results constantly and while it does not find the appropriated knowledge to achieve a certain goal, it will pursue a better KO.

#### 5. Related Work

In most of the agent platforms, when an agent cannot deal with an unknown event or does not have the knowledge required to achieve a certain goal, the event or goal are simply dropped. However, some platforms already consider the use of a “default knowledge” that is used when the knowledge available cannot be applied to achieve a specific

goal (for instance, the implementation of 3APL [11] assumes the use of a “default plan” to deal with an event without suitable plans associate to it).

In [4], it is presented the Coo-BDI approach. This approach aims to implement agents that dynamically change their behaviors by cooperating with other agents (“cooperation” means to retrieve external plans for achieving certain desires). Although this work has some similarities with ours, in a broader view there are some important differences that must be taken into account. Firstly, we believe that an action plan just represents a fragment of the necessary knowledge for an agent to achieve a goal. For instance, to execute properly a plan (and consequently to achieve a certain goal) it is necessary for a BDI agent to have specific beliefs. These beliefs should be retrieved together with the plan. Thus, we indicate the use and exchange of knowledge objects instead of only action plans. Other significant difference of our proposal is the execution history usage. The COO-BDI approach focus on exchanging plans to cope with the situation in which no local plans are available for managing an event or goal. [5] shows theoretically how the Coo-BDI approach can be used by AgentSpeak’s agents implemented with Jason.

Other proposals partly investigate the integration of KM processes in MAS organizations. Just to make some examples, [12] presents an architecture to design computational intelligence systems including a framework for KM. This paper covers some aspects of a KM process but they are rather superficial. For instance, it does not present the knowledge representation format to exchange knowledge. In [13], issues related to the knowledge capture and distribution are addressed. Though this work considers these two issues, several other issues are missing. There is no explanation about the knowledge selection mechanism or about the degree of compatibility of a specific knowledge and the agent’s needs.

## 6. Final Remarks and Future Works

This paper presented general aspects of the SemantiCore+Ontowledge (S+O) framework. S+O focuses on knowledge distribution and use. The main knowledge organization unit is the knowledge object (KO) which encapsulates all necessary items to achieve a specific goal. KOs are represented as ontologies for knowledge sharing.

We believe that the ability of exchanging knowledge represents a considerable improvement in the development of MAS. The benefits of the proposed architecture also can be considering in a pervasive environment. An important requirement for an agent in a pervasive environment is the support for configurable and adaptive behavior in order to dynamically react to the changes in the environment and in the agent’s goals. To satisfy such requirement, we need an infrastructure with knowledge management features.

S+O has some limitations and improvements opportunities. For instance, a visual agent development tool

must be provided in order to facilitate the agent definition. This visual composer may also have intrinsic ontology development support so the facts and rules can be directly defined and tested using ontologies. An IDE such as Eclipse can be adapted to help KOs creation together with the regular application code.

## Acknowledgments

Study developed by the Intelligent Systems Engineering Group of the PUCRS, financed by Dell Computers of Brazil Ltd. (addendum PDTI) with resources of Law 8.248/91.

## References

- [1] T. Berners-Lee, J. Hendler, and O. Lassila. “The Semantic Web”. *Scientific American Magazine*, May 2001.
- [2] L. Kagal *et al.* “Agents Making Sense of the Semantic Web”. In: Proc. First GSFC/JPL Workshop on Radical Agent Concepts, 2002, pp. 417-433.
- [3] J. Elin. “Ontologies and the Semantic Web”, *Bulletin of the American Society for Information Science & Technology*, Apr/May 2003.
- [4] D. Ancona and V. Mascardi. “Coo-BDI: Extending the BDI model with cooperativity”. In: Proc. of DALI-03, 2003.
- [5] D. Ancona *et al.* “Coo-AgentSpeak: Cooperation in AgentSpeak through Plan Exchange”. In: Proc. of AAMAS’2004, 2004, pp. 698–705.
- [6] T. Kucza, “Knowledge Management Process Model”. <http://www.inf.vtt.fi/pdf/publications/2001/p455.pdf>, 2005.
- [7] A. Macintosh, I. Filby, and A. Tate. “Knowledge Asset Road Maps”. In: Proc. of PAKM98. Basel, 1998.
- [8] J. Stader and A. Macintosh. “Capability modelling and knowledge management”. *Applications and Innovations in Intelligent Systems VII*, Springer-Verlag, 1999, pp 33-50.
- [9] M. Obitko and V. Marik. “Ontologies for Multi-Agent-Systems in Manufacturing Domain”. In: Proc. of the 13th International Workshop on Database and Expert Systems Applications, France, 2002, pp. 597- 602.
- [10] A. Tello and A. Gómez-Pérez. “ONTOMETRIC: A Method to Choose the Appropriate Ontology”. *Journal of Database Management*, 15, 2 (2004), pp. 1-18.
- [11] 3 APL platform. <http://www.cs.uu.nl/3apl/>, 2008.
- [12] R. Weber and D. Wu. “Knowledge management for computational intelligence systems”. In: Proc. of the 8th IEEE International Symposium on High Assurance Systems Engineering (Singapore, 2004), pp. 116–125.
- [13] H. Rybinski and D. Ryzko. “Knowledge sharing in default reasoning based multiagent systems”. In: IEEE/WIC IAT’03, (Halifax, 2003), pp. 576-579.

# JAAF: A Framework to Implement Self-Adaptive Agents

Baldoino F. dos S. Neto<sup>1</sup>, Andrew D. da Costa<sup>1</sup>, Manoel T. de A. Netto<sup>1</sup>, Viviane T. da Silva<sup>2</sup>  
and Carlos J. P. de Lucena<sup>1</sup>

<sup>1</sup>PUC-Rio, Computer Science Department, LES – Rio de Janeiro – Brazil

<sup>2</sup>Universidade Federal Fluminense, Computer Science Department – Niterói – Brazil

{bneto, acosta, mnetto, lucena}@inf.puc-rio.br  
viviane.silva@ic.uff.br

## ABSTRACT

*The self-adaptation paradigm aims to develop software systems that can autonomously adapt themselves to context changes and handle adverse situations on their own. However, appropriate implementation of self-adaptive software systems is still an open issue. Therefore, this paper proposes a framework (Java self-Adaptive Agent Framework - JAAF) to implement self-adaptive agents based on a set of steps to perform self-adaptations of software agents. The framework provides support to three main agent-related properties: autonomy, pro-activity and reasoning. Besides taking advantage of software agents to better implement self-adaptive software systems, JAAF extends the JADE framework that already gives support to autonomy and pro-active agents. In order to provide reasoning agents, JAAF uses a set of methods based on rules, cases and genetic algorithms.*

## 1. INTRODUCTION

The complexity of current systems has influenced the software engineering community to look for systems able to adjust or adapt their behavior in response to changes in the environment. In this context, autonomic computing and consequently self-adaptive systems have become one of the most promising directions.

Although there are several approaches [1]-[3] describing how systems can perform self-adaptations, they are not implemented by the use of software agents, and therefore do not contemplate agent-related properties that are important to self-adaptive systems, such as autonomy, learning, reasoning and pro-activity [5], [6]. Besides, they do not provide the adequate flexibility to create different plans for self-adaptation. Such flexibility is important to make it possible to implement different processes that promote self-adaptability in different domains.

Therefore, the Java self-Adaptive Agent Framework (JAAF) was proposed. In order to take advantage of software agents to better implement self-adaptive software systems, such framework extends the JADE framework that

already gives support to autonomy and pro-active agents. By using JAAF it becomes easier the construction of self-adaptive agents due the infra-structure provided by the framework, which offers support to the implementation of different self-adaptation processes composed of activities that can perform collect of data, analysis, decisions, etc. The framework provides mechanisms that help the implementation of such activities, such as, reasoning methods based on rules, cases, genetic algorithms, besides different ways of collecting data, format them, etc. JAAF can be instantiated to implement, for instance, biological, ubiquitous computing and autonomic computing systems.

The paper is organized as follows. Section 2 presents some related work. In Section 3 the JAAF is detailed and Section 4 states a case study by describing how agents can perform self-adaptation in real situations. Finally, Section 5 concludes and presents some future work.

## 2. RELATED WORKS

In [1] the authors propose a self-adaptation process based on four key activities: collect, analyze, decide, and act. The collect activity provides mechanisms that collect, aggregate, filter and report details (such as metrics) collected from an application. The analyze activity correlates and models complex situations aiming to detect problems. The decision activity provides mechanisms that construct the actions that the agents need to execute in order to achieve goals. And finally, the act activity controls the execution of the actions proposed by the decision component. The main difference between our approach and theirs is that our framework can be used to instantiate different self-adaptation processes composed of any set of activities.

Rainbow [2] uses an abstract architectural model at runtime to monitor the properties of an executing system, to evaluate the model for constraint violation, and — if a problem occurs — to perform global and module adaptations on the running system. Rainbow uses mechanisms based on fix adaptation plans to monitor and

adapt the system behavior at runtime and uses constraints (rule) in order to verify problems or violations. Besides this, it uses utility functions [15] to determine the most appropriate adaptation within a set of applicable ones. Aiming to propose an approach more flexible, the JAAF enables the elaboration of different adaptation plans and provides not only rule-based reasoning, but also case-based reasoning (CBR) mechanisms. According to [8], CBR is an efficient way to implement some of the properties of autonomic systems.

Kinesthetics extreme (KX) [3] works on the implementation of a complete autonomic loop. This work was driven by the problem of adding autonomic properties to legacy systems; that is, existing systems that were not designed with autonomic properties in mind. However, sometimes it is not possible to modify these systems. Thus, the addition of autonomic properties is required to be completely decoupled from the system with autonomic monitoring sensors added on the top of existing system and monitoring functionality. KX focuses on the collection and processing of monitoring data from legacy systems and execution of repair. It does not provide support to the elaboration of different adaptation plans and does not have the intention of providing different reasoning mechanisms.

The Agent Building and Learning Environment (ABLE) [4] provides an autonomic management in the form of a multi-agent architecture; that is, each autonomic manager is implemented as an agent or set of agents, thus allowing different autonomic tasks to be separated and encapsulated into different agents. Although ABLE provides reasoning mechanisms, it does not allow the elaboration of different self-adaptation plans.

### 3. JAAF

In this section we describe the main idea of the framework, followed by the analysis of the JAAF class diagram and the details about its kernel (frozen-spots) and flexible points (hot-spots) [13].

#### 3.1 MAIN IDEA

The JAAF framework is implemented by the use of software agents, as illustrated in Figure 1 by the JAAF architecture. JAAF extends JADE [9], [10], a FIPA compliant framework to implement multi-agent systems (MAS) developed in Java, in order to represent three concepts: (i) agents that perform self-adaptation, (ii) plans executed by agents representing self-adaptation processes (or control loops) and (iii) activities that are the steps of such processes. In order to implement self-adaptive systems it is necessary to instantiate the JAAF framework by implementing the plans or control loops and their activities. JAAF already provides a control loop composed of four activities, as detailed in Section 3.2.

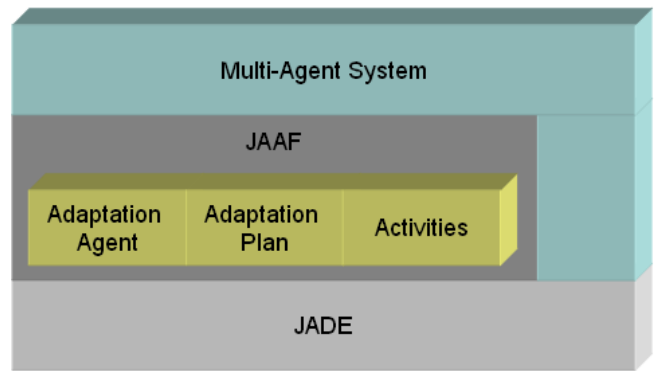


Figure 1. JAAF Architecture

#### 3.2 DETAILS OF THE JAAF

The class diagram depicted in Figure 2 illustrates the main JAAF classes. The self-adaptive agents are represented by the *AdaptationAgent* class and the self-adaptation process by the *PlanAdaptation* class. Such class extend the JADE class *FSMBehaviour* that provides support to the implementation of finite automata composed of activities (or behaviors) represented by the *Behaviour* class.

JAAF already provides a self-adaptation process represented by the *ControlLoop* class that is composed of four activities (Figure 3): Collect (Section 3.2.1), Analyze (Section 3.2.2), Decision (Section 3.2.3) and Effector (Section 3.2.4), each one extending the *Behaviour* class. This control loop was created based on the self-adaptation process mentioned in [1]. Note that, although JAAF already provides such control, it is possible to define others from the *PlanAdaptation* class. It is also possible to implement different activities (or steps of the loop), from the *Behaviour* class, to different control loops.

##### 3.2.1. COLLECT

This is the first step executed by the process. It is responsible for providing mechanisms to collect, aggregate and filter (format) data collected from the application.

In order to represent this idea, the framework offers two concepts: sensor and format. The *sensor* defines the place where the data should be collected (database, log, etc) and the *format* defines the format of the collected data. This activity also specifies *when* the agents should collect the data, i.e., it describes the preconditions to activate the sensor.

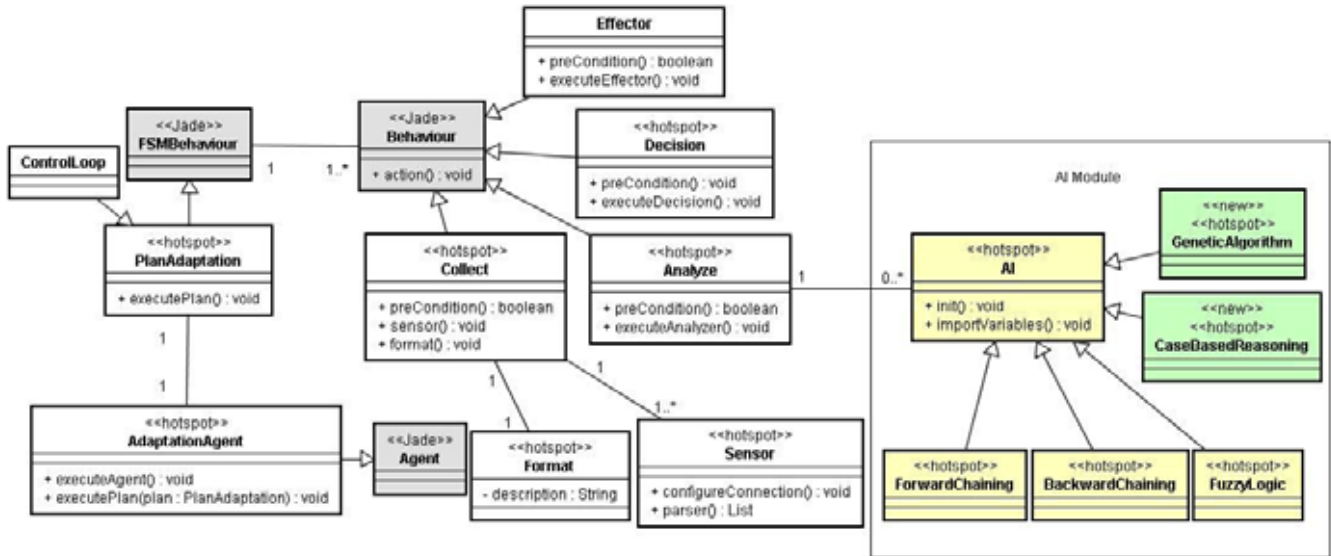


Figure 2. Class Diagram of the JAAF

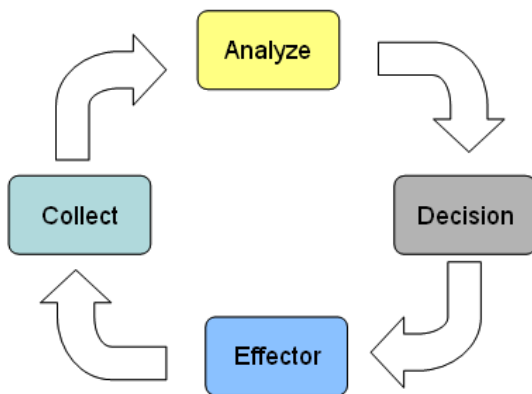


Figure 3. Control loop provided by the JAAF framework

### 3.2.2. ANALYZE

The analyze activity is responsible for providing mechanisms that analyze the data collected in the previous activity in order to detect problems and suggest new solutions. The framework gives support to three techniques: (i) rule based reasoning (forward chaining, backward chaining and fuzzy logic) [12], case based reasoning [8] and genetic algorithm [11].

### 3.2.3. DECISION

Decision is the third activity responsible for deciding which action (or behavior) will be the next one to be executed by the agent, while trying to achieve the goal. Such decision is based on the information provided by the previous step.

### 3.2.4. EFFECTOR

This is the last step of the self-adaptation process. It receives the selected action from the *Decision* activity, and informs the agent the action to be executed. When the action is executed, the control loop can be executed again whether any self-adaptation is necessary.

## 3.3 HOT-SPOTS AND FROZEN-SPOTS

Since JAAF extends JADE, the JADE kernel is also the kernel of JAAF and the hot-spots of the JADE are the hot-spots of JAAF. For instance, the process used by agents to communicate, and the agents' identifiers are examples of JAAF hot-spots inherited from JADE.

The hot-spots specifically defined in JAAF are:

1. Agent (*AdaptationAgent* class): By extending such class and implementing the *executedPlan* method, it is possible to define different algorithms to execute the plans of an agent.
2. Plan of self-adaptation (*PlanAdaptation* class): It is possible to define new control loops (or plans) and the sequence to execute the activities of the control loops. JAAF already provides a default control loop implemented in the *ControlLoop* class.
3. Activities (*Behaviour* class): It is possible to define new activities to be called by the control loops by extending the *Behaviour* class. JAAF already offers four activities (Collect, Analyze, Decision and Effector).
4. Sensor (Sensor class): One can define *when* and *where* the data should be collected.
5. Format (Format class): It is possible to define the

format of the data to be collected by the Sensor.

- Intelligent Algorithm module: JAAF offers three kinds of algorithms: rule-based reasoning (forward chaining, backward chaining and fuzzy logic), case-based reasoning and genetic algorithm. Such algorithms can be used at any point of the system to help with the self-adaptation.

As mentioned above, in order to implement a self-adaptive agent, the following steps should be performed: (1) define a plan of self-adaptation by extending the *PlanAdaptation* class; (2) create the activities that compose such plan by extending one of the JADE classes that represent behaviour; and finally (3) create a self-adaptive agent by extending the *AdaptationAgent* class.

#### 4. CASE STUDY: CREATION OF SUSCEPTIBILITY MAPS

Landslides are natural phenomena, which are difficult to predict since they depend on many (unpredicted) factors and on relationships among those factors. The annual number of landslides is in the thousands, and the infrastructural damage is in the billions of dollars [14]. Since there is a need to systematically deal with these factors, one of the main challenges faced by the specialists is to decide the most appropriate model configuration to generate susceptibility maps (SM), i.e., maps that show locations with landslides risks in a specific area. By using such a map, it is possible to identify the areas with highest risks in a region.

In this context, we used the JAAF framework to create a multi-agent system in order to generate an SM that shows the places with landslide risks of Rio de Janeiro, a city in Brazil. Each application agent is able to *adapt* the configuration of its *susceptibility model* in order to meet the SM closest to the one that represents the reality.

##### 4.1 MAIN IDEA

The implemented system is composed of three agents, as illustrated in Figure 4. The goal of the two susceptibility generator agents (SGA) is to meet the most appropriate configuration of its susceptibility models. The most appropriated configuration is found by comparing the SM – generated by using the susceptibility model and its configuration — with an inventory map, which is a map that stores the history of landslides that have occurred in Rio de Janeiro over the last thirty years.

In this example, both agents use the same susceptibility model but use different types of data and reasoning algorithms (case-based reasoning and genetic algorithm) to perform the self-adaptations.

When each SGA finds its most appropriate configuration, it provides such configuration to the decision agent. When the decision agent receives both configurations, it configures the model and generates two

SMs. It compares the two SMs in order to discover which is closest to the landslide history. Note that the decision agent defines a timeout to receive such configurations.

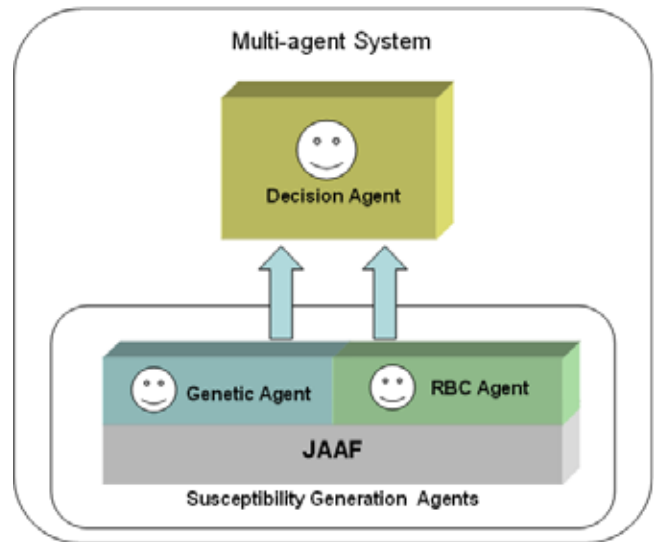


Figure 4. Conceptual model of the case study

##### 4.2 SUSCEPTIBILITY GENERATION AGENTS

The model used by both SGAs is described in equation 1.  $S$  represents the susceptibility value of a specific area in an SM,  $X$  represents the factors considered to generate the susceptibility value and  $W$  is the weight of the type of such factors.

$$S = 1 - \sum_{i=1, n} W_i * X_i \quad (1)$$

Three types of factors (types of data) can be used: Vegetation type (V), Slope (S) and Accumulated Rain (R).

Tables 1, 2 and 3 present examples of different types of vegetation, different slope percentage and degrees of accumulated rain together with the scale that should be considered in each case. For instance, consider an area that is constituted by a preserved forest, which has a slope of 5% and accumulated rain of 65 mm. The scales to be considered in such a case are 0.85, 0.75 and 0.15. The vegetation and the slope indicate that it is a preserved area with a low slope and that the risk of landslides in such an area is low. But the accumulated rain drastically increases such risk.

Table 1: Vegetation Scale Configuration

Vegetation	Scale
Preserved Forest	0.85
Degraded Forest	0.65
Uncovered Soil	0.0
Grass	0.35
Plantation	0.10
Floodplain	0.15

Table 4 exemplifies the weights attributed to each type of factor. In the example, the accumulated rain is a little bit



more important than the other two types of factors when evaluating the landslide risk.

**Table 2: Slope Scale Configuration**

Slope	Scale
(0-3)%	0.9
(4-8)%	0.75
(9-25)%	0.6
(26-45)%	0.4
(46-75)%	0.2
(>75)%	0.0

**Table 3: Accumulated Rain Configuration**

Accumulated Rain (mm)	Scale
(0-10)	0.90
(11-20)	0.84
(21-30)	0.71
(31-40)	0.53
(41-60)	0.53
(>60)	0.15

**Table 4: Types of Factors Weights**

Type of Factor	Weight
Vegetation Type	0.3
Slope	0.3
Accumulated Rain	0.4

It is the goal of the two SGAs to choose the set of types of factors to be used to generate the SM, to define the scales exemplified in Tables 1, 2 and 3 and also to state the weights that should be considered in each type of factor. By adapting the parameters of equation 1, the agents try to provide the most appropriate configuration.

#### 4.2.1. GENETIC AGENT

While instantiating JAAF to implement the *genetic agent*, it was not necessary to define a new control loop or new activities. The control loop provided as default was used and the four activities already identified were implemented.

In the collect activity the agent collects the configuration used to generate the actual SM, data from the inventory map and formats them in an adequate format that can be manipulated by the others activities of the control loop. The JAAF was useful due to the available mechanisms that allow to read and format data of different sources. Next, in the analyze activity the agent uses a genetic algorithm to analyze the collected data and the actual configurations used to generate the previous SMs. After such analyses, a set of configurations is suggested for the decision activity; i.e., in the analyze activity the agent *adapts* the parameters of equation 1 aiming to meet better configurations than those previously used.

The decision activity uses a rule-based reasoning algorithm to meet the ideal configuration of a susceptibility

model while considering only the vegetation type and slope data. Since both the genetic algorithm and the rule-based reasoning algorithm are provided by JAAF, the instance only needed to provide the necessary parameters to execute such algorithms.

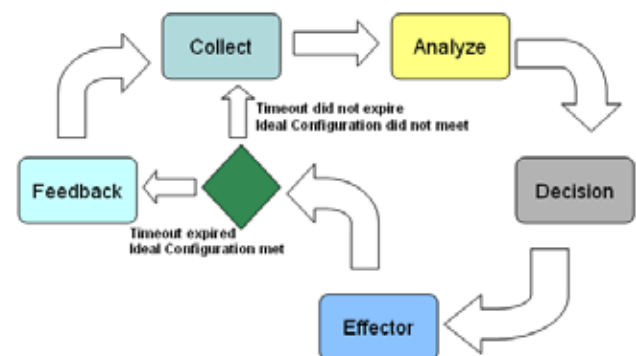
In the last activity called effector, the agent uses equation 1 with the configurations suggested in the previous activity to generate the SM. Note that the control loop can be executed repeatedly until the agent meets the most appropriate configuration or until the timeout defined by the decision agent expires.

#### 4.2.2. RBC AGENT

As the *Genetic agent*, *RBC agent* is also an SGA that applies self-adaptation in order to meet an ideal configuration of the scales and weights. However, the control loop used is composed of five activities (Figure 5): Feedback, Collect, Analyze, Decision and Effector. Since the framework provides mechanisms that help on the implementation of such activities and on the communication between them, it is not difficult to implement the needed control loop.

In the collect activity the agent retrieves data from the inventory map, the configuration used to generate the actual SM and formats them to be manipulated by the case-based reasoning mechanism that is used in the next step. The analyze activity uses a case-based reasoning algorithm in order to analyze the collected data from the collect activity and the cases stored in a case-base (past experiences), aiming to suggest a set of better configurations than those previously used. In this case, the instance only needed to provide the necessary parameters to execute the case-based reasoning algorithm, since such algorithm is already provided by JAAF.

The decision activity uses a rule-based reasoning algorithm on the configurations suggested in the previous step in order to meet the ideal configuration of a susceptibility model while considering the vegetation type, slope and accumulated rain data.



**Figure 5. Control loop defined by the RBC agent**

The next activity, called effector, receives the met configurations in the previous activity and executes them. If the timeout defined by the decision agent expires or the

best configuration is met, then the *RBC* agent sends the configurations to the decision agent and the feedback activity is executed. Otherwise, the collect activity is executed.

The feedback activity is responsible for receiving the chosen configuration by the decision agent and stores it in a case-base as a successful case. This task contributes to expanding agent learning.

## 5. CONCLUSION AND FUTURE WORKS

This paper proposes a framework that provides support to the construction of self-adaptive agents by the implementation of different plans for self-adaptation. Nonetheless, it also provides reasoning methods based on rules, cases and genetic algorithms that can be used by the agents.

The applicability of such a framework can be verified by the case study presented in Section 4. The two agents illustrated in the section use different self-adaptation processes. One of them uses the process proposed as default by the framework – the control loop composed of four activities – while the other instantiates the framework by implementing another activity and defining a different self-adaptation process. In addition, these agents are able to self-adapt the configurations of the susceptibility model used to generate SMS by using different reasoning mechanisms proposed by the framework.

We are in the process of defining new self-adaptation control loops and mechanisms able to handle several of them, not only control loops but also complex context using ontology. It is also our intention to extend JAAF in order to provide a framework not only for self-adaptation but also for self-organization in a multi-agent environment. Such a framework would guide the development of organizations inspired by biological systems.

## REFERENCES

- [1] S. Dobson, S. Denazis, A. Fernández, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Transactions Autonomous Adaptive Systems (TAAS)*, 1(2):223-259, December 2006.
- [2] David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl and Peter Steenkiste. Rainbow: Architecture-Based Self Adaptation with Reusable Infrastructure. In *IEEE Computer*, Vol. 37(10), October 2004.
- [3] Kaiser, G.; Parekh, J.; Gross, P. and Valetto, G. Kinesthetics eXtreme: An external infrastructure for monitoring distributed legacy systems. In *Proceedings of the Autonomic Computing Workshop at the 5th Annual International Workshop on Active Middleware Services (AMS)*, 2003.
- [4] Bigus, J. P.; Schlosnagle, D. A., Pilgrim, J. R.; et. al.. ABLE: A toolkit for building multiagent autonomic systems. *IBM Syst. J.* 41, 3, 350–371, 2002.

- [5] Jennings, N. R. and Wooldridge, “M. Agent-oriented software engineering,” In Bradshaw, J. (Ed.) *Handbook of Agent Technology*, AAAI/MIT Press, 2000.
- [6] Wooldridge, M. and Jennings, “N. R. Pitfalls of agent-oriented development,” *Proceedings of the Second International Conference on Autonomous Agents (Agents’98)*, ACM Press, pp. 385-391, 1998.
- [7] Huebscher, M. C. and McCann, J. A. A survey of Autonomic Computing—Degrees, Models, and Applications. *ACM Computing Survey*, August 2008.
- [8] A. Amodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. In *AI Communications*, volume 7:1, pages 39–59. IOS Press, March 1994.
- [9] Bellifemine, F., Caire, G., Trucco, T., Rimassa, G., Mungenast, R., *Jade Administrator’s Guide*, 2007.
- [10] Bellifemine, F., Caire, G., Trucco, T., Rimassa, G., *Jade Programmer’s Guide*, 2007.
- [11] Melanie Mitchell, *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, The MIT Press, February 6, 1998.
- [12] Costa, A., Lucena, C. J. P.; Silva, V., Cowan, D.; Alencar, P., A Hybrid Diagnostic-Recommendation System for Agent Execution in Multi-Agent Systems, *ICSOFT 2008 – 3<sup>rd</sup> International Conference on Software and Data Technologies*, Porto, Portugal, July 2008.
- [13] Fayad, M. , Johnson, R., *Building Application Frameworks: Object-Oriented Foundations of Framework Design (Hardcover)*, Wiley publisher, first edition, ISBN-10: 0471248754, 1999.
- [14] Karim S. Karam, *Landslide Hazards Assessment and Uncertainties*, Thesis: Massachusetts Institute of Technology, September 2005.
- [15] Petrucci, V. and Loques, O. Suporte a adaptação de aplicações usando funções de utilidade. In *1st Workshop on Pervasive and Ubiquitous Computing, WPUC 2007, SBAC-PAD 2007*, October 2007.

# An Agent-based centralized e-Marketplace in a Virtual Environment

Ingo Seidel, Markus Gärtner,  
Josef Froschauer, Helmut Berger  
Matrixware Information Services GmbH,  
Lehargasse 11/8  
A-1060 Vienna, Austria  
Email: {firstname.lastname}@matrixware.com

Dieter Merkl  
Institute of Software Technology and Interactive Systems  
Vienna University of Technology  
Favoritenstrasse 9-11/188  
A-1040 Vienna, Austria  
Email: dieter.merkl@ec.tuwien.ac.at

## Abstract

*In this paper we present the design and implementation of an agent-mediated B2C e-Marketplace as part of the social and immersive 3D e-Tourism environment “Itchy Feet”. Customers are able to interact in an intuitive way in the 3D environment and assess tourism products prior to purchase. The e-Marketplace provides services via autonomous software agents for the entire purchase cycle. Fair trade among agents is ensured by regulating the environment with Electronic Institutions.*

## I. Introduction

Electronic Marketplaces (EMs) are electronic platforms connecting buyers and sellers to conduct business. Wang et al. [10] did a comprehensive literature review of EM research and ascertained that most economic researchers studied decentralized electronic markets but paid less attention to central platforms. Fisher and Craig [5] discovered that the lack of social interaction in online channels and the conflict between online and traditional channels are crucial issues that hinder the adoption of EMs. Furthermore it is important to enforce security mechanisms within a marketplace to ensure fair trade. To address these aspects our work concentrates on the creation of a central platform in the B2C domain. The platform consists of a Multi-Agent System to form and regulate the marketplace, a virtual environment to encourage user participation and communication facilities to support the formation of a community. The whole marketplace is modeled in the virtual environment. We intend to leverage 3D visualization for the presentation of products within the virtual environment. This opens new possibilities for providers to present their products and consumers are able to get more insights than with text and picture only presentations. To establish a sustainable

community we provide a common meeting place for users to communicate and interact with each other. To encourage trust and security we apply Electronic Institutions (EIs) to regulate the marketplace. Electronic Institutions are a Multi-Agent System methodology and provide a facility for defining and regulating interactions between software agents [4]. In our previous work we have connected EIs to a 3D Virtual World to form the 3D e-Tourism environment “Itchy Feet” [2], [7]. Itchy Feet has been developed as part of a project in the tourism domain with the principal goal of creating a 3D e-Tourism environment to support the complex interaction patterns of providers and consumers in e-Tourism. Autonomous software agents are used to render the environment information rich and EIs are used to regulate the actions of all participants. Software agents and users participate in the system and are visualized in the 3D Virtual World.

In this paper the final trading mechanisms of Itchy Feet are presented and special focus is placed on the realization of the auction process. The remainder of this paper is structured as follows. In Section II the related work is presented. Electronic Institutions and the Itchy Feet project are presented in Section III. The marketplace is presented in Section IV and Section IV-B illustrates how auctions are conducted in the 3D Virtual World. In Section V the paper is concluded and an outlook on future work is given.

## II. Related Work

Mavetera and Kadyamatimba created a conceptual framework for agent-mediated trading in e-Markets [6]. They identified several stages and components such as product brokering, negotiation and contract formation that are needed to implement a full e-Market system using agent technology. Wang et al. [10] did a comprehensive literature review on the current EM research. They identified eight major research themes and showed that most

studies address only the potential but not the real impact of EMs. Furthermore, the results indicated that most of the research methodologies are rather “qualitative” than “quantitative”.

3D Virtual Worlds have been used in research for the past 10 years and several researchers have worked on the more specific topic of combining Multi-Agent Systems and 3D Virtual Worlds. Smith et al. [8] present an approach where the agent logic is incorporated in a 3D environment. According to the authors most worlds are largely static and objects are used to trigger pre-programmed behavior. Agents are supposed to enrich the world and should make the environment more dynamic. The proposed framework consists of a society of agents in which each agent controls a 3D object. Adobbati et al. [1] present GameBots; a system that abstracts from the Multi-Agent System and provides a uniform interface to the 3D Virtual World. The created environment is a multi-agent research test bed that is not limited to a specific task in a fixed environment and supports human testing and interaction. Traum & Rickel [9] studied dialogue models between humans and software agents in 3D Virtual Worlds. They concentrated on issues such as proximity and attentional focus of others, the interplay between speech and nonverbal signals and the ability to maintain multi-part conversations.

### III. Framework Description

In Itchy Feet, Electronic Institutions are connected to a 3D Virtual World in order to allow human users to participate in the agent system and to enable the visualization of software agents in the 3D Virtual World. The concept of EIs as well as the framework connecting them to the 3D Virtual World are presented in the following.

#### A. Electronic Institutions

Electronic Institutions resemble real-world institutions by using formal specifications to define interaction patterns between agents [4]. These specifications describe what agents are eligible to do within an EI. The building blocks of EIs are i) the *Dialogical Framework*, ii) the *Performative Structure*, and iii) the *Norms* and behavioral rules. The *Dialogical Framework* defines the ontology and social structure within the EI. The *Performative Structure* comprises scenes and transitions. Every scene has a protocol that defines the possible interaction patterns among agents within that scene. Scene protocols are defined as finite state machines. A state change is performed when an agent utters a message or a timeout occurs. It is possible to define which agents are eligible to join or leave the scene in which states. Scenes are connected by transitions. Whenever an agent leaves a scene it needs to traverse a

transition to get to the next scene. Transitions may impose restrictions on the movement of an agent. Norms, i.e. the behavioral rules, establish role-based conventions that are used to verify if interacting agents behave according to the system’s normative specification. A detailed description of EIs is given in [4].

#### B. 3D Electronic Institution Framework

In Itchy Feet two types of participants need to be considered: humans and agents. Agents are either autonomous or controlled by a human user. In the latter case, the couple human/agent is represented as an avatar in the 3D Virtual World. The user delegates tasks such as information gathering or product purchasing to the agent and learns from the agent which rules and restrictions apply in the environment. The user must act according to these rules. The movement and actions of the user in the 3D Virtual World are verified by the agent in the EI. Autonomous agents must be visualized in the 3D Virtual World such that users are able to interact and learn from them. The visualization of autonomous agents depends on their task. For example, an agent that actively participates in conversations may be visualized as an avatar, whereas a simple information agent may be visualized as an information monitor. The dependence between the two systems requires that the 3D Virtual World is causally connected to the EI. In our case this means that whenever the 3D Virtual World changes, the EI must change as well. Whenever the EI evolves, the 3D Virtual World has to be modified in order to maintain a consistent relationship. Conceptually speaking, the system is composed of three layers. The 3D Virtual World is located at the top layer, the EI is located at the bottom layer and both components are causally connected by the middleware [3], [7]. A detailed description of the system architecture can be found in [7].

### IV. The Itchy Feet Marketplace

Electronic Institutions are a useful framework for the creation of an agent-based e-Marketplace. They enable the participation of heterogeneous autonomous software agents and define a regulatory environment that governs the actions of these agents. However, the ability for human users to engage in EIs is rather limited. In order to overcome these limitations the 3D Virtual World is used as an alternative user interface for end users. The fact that the user is participating in the Multi-Agent System is hidden by the framework and the user is only presented with those interface controls that are necessary to complete the user’s goals. The 3D Virtual World of Itchy Feet is composed of three buildings which offer various services to the users. The product trade takes place in the Travel

Agency and Auction House buildings. Products are traded in two different ways: in the Travel Agency the user buys products for a fixed price while in the Auction House the user purchases a product within an auction. Every building is a separate sub-marketplace and the product purchase needs to be completed within the boundaries of this sub-marketplace. For example, the user is only allowed to leave the building if all the products in the user's shopping cart have been paid. This mechanism enables the support of multiple sub-marketplaces that can be defined by different providers to suit the needs of each individual provider. The third building, the Forum building, is the meeting place of Itchy Feet. In this building users are able to hang out, they can share their knowledge and they are able to get help from expert users. An overview of the 3D Virtual World with the three buildings, including a detailed picture of the Auction House, is shown on the left side of Figure 1. The functionality which is available in these buildings is defined in the Multi-Agent System. Every building corresponds to exactly one Electronic Institution. The scenes of an EI are visualized as rooms inside the building and the transitions are visualized as doors connecting these rooms. Whenever the user moves around in the 3D Virtual World, the user's agent mimics the movement in the EI. Since there is no possibility to regulate inter-Electronic Institution communication, a separate EI named *Ether* has been designed. In contrast to the other EIs, the *Ether* is not mapped to a building in the 3D Virtual World. The *Ether*'s functions are accessible at every location in the 3D Virtual World and include the shopping cart, the inventory and the chat. Note that agent role names are henceforth formatted in italic. The term *User* refers to the role in the EI that can be played by a software agent as well as a human user. In contrary the term "user" refers to an actual human user that is participating in Itchy Feet via the 3D Virtual World.

### A. The Auction Protocol

The Auction House EI along with the Travel Agency EI constitute the e-Marketplace of Itchy Feet. The Auction House comprises six scenes: the offering scene, the information scene, the clearing scene and three auction scenes. The offering scene, lead by the *OfferManager*, is the control unit of the Auction House. The *OfferManager* overlooks the commodity flows and ensures that every product is put on auction at the scheduled time. *Users* inform themselves about available products in the information scene and pay for auctioned products in the clearing scene. Two types of agents, namely *User* and *Auctioneer*, participate in the auction scene. The possible interactions between these agents are determined by the auction scene protocol.

Prior to the auction start, the *OfferManager* hands over the product to the *Auctioneer*. As soon as the *Auctioneer* receives the product, it informs all agents in the scene that the auction will start. The process is based on a real world English auction, resembling a forward auction starting at a low price where bidders raise the price until no further bids are issued or the bidding time has exceeded. Just like in a real world auction the *Auctioneer* announces the three different states "going once", "going twice" and "sold". An auction ends in one of three different ways: i) a bid has been issued and the *Auctioneer* announces "going once", "going twice" and "sold" without another bid from a different *User* being placed, ii) the bidding time has elapsed making the last *User* which issued a bid the winner, iii) the bidding time has elapsed and no bids have been issued. In any of these cases the *Auctioneer* informs the *OfferManager* whether the product was sold or not. After an auction has ended the *Auctioneer* remains in a waiting state until it receives another product from the *OfferManager*.

### B. Auctions in the 3D Virtual World

The auction scene is visualized as a separate room of the Auction House in the 3D Virtual World. The auction room as well as the auction interface are shown on the right side of Figure 1. The auction interface shows the item to be auctioned, the current status of the auction and contains an input box where the next bid is entered. The screenshot shows four actors which are participating in the auction. Two of them are autonomous software agents and two of them are human users who are logged in the 3D Virtual World. The autonomous software agents are visualized by the 3D Virtual World. The roles of these agents determine how and where they are visualized. The *Auctioneer* agent, which is responsible for conducting the auction, is visualized on the stage behind the podium. The other autonomous agent is playing in the *User* role and is also interested in buying the product. This agent is visualized among the other users and is wearing the robot-like outfit. The different locations and outfits of each agent role help the user to quickly identify the duties of each avatar and make it easier to differentiate the individual avatars.

The upcoming auctions are displayed on an information panel in the information room of the Auction House. When a user decides to participate in an auction, she enters the auction room. As a consequence the user's agent enters the auction scene in the EI. The *Auctioneer* starts the auction at the given time following the auction protocol. When a bid is submitted by the user, a request is sent to the user's agent, which sends out a bid message in the auction scene of the Auction House EI. The actions of each



Fig. 1. The 3D Virtual World with the Auction House building and the Auction Room

user and agent are visualized in the 3D Virtual World by gestures and other visual cues. The bidding is illustrated by a hand raising gesture and by a message box with the bid amount that pops up over the avatar. If a user is announced the winner of the auction, the product is displayed in the shopping cart in the 3D Virtual World. The product is then to be paid in the clearing room where the clearing scene is visualized.

## V. Conclusion & Future Work

In this work we have utilized 3D Virtual Worlds as a new type of user interface for agent-based e-Marketplaces. The focus was placed on the creation of an easy to use interface enabling end users to participate in Electronic Institutions and to interact with software agents. The marketplace in this environment is realized by means of auctions and fixed price product trade. The trade processes are hereby defined in the EI and autonomous software agents are responsible for their execution. In particular, we have illustrated how auctions are conducted, how the user is able to participate in a natural way and how the connection between the user and the EI works.

The developed system will be used as a test-bed for studying the interaction and behavior of users in an e-Market setting. The next steps of our work involve the execution of user studies where the acceptance of the environment is evaluated.

## VI. Acknowledgments

This work was funded by the FWF Austrian Science Fund (project reference: L363).

## References

- [1] R. Adobbati, A. N. Marshall, A. Scholer, S. Tejada, G. Kaminka, S. Schaffer, and C. Sollitto. Gamebots: a 3d virtual world test-bed for multi-agent team research. *Communications of the ACM*, 45(1):43–45, 2002.
- [2] H. Berger, M. Dittenbach, D. Merkl, A. Bogdanovych, S. Simoff, and C. Sierra. Opening new dimensions for e-tourism. *Virtual Reality*, 11(2):75–87, 2007.
- [3] A. Bogdanovych, H. Berger, C. Sierra, and S. Simoff. Humans and agents in 3d electronic institutions. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 1093–1094, New York, NY, USA, 2005. ACM Press.
- [4] M. Esteva, J. Rodriguez-Aguilar, C. Sierra, P. Garcia, and J. Arcos. On the formal specifications of electronic institutions. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective*, pages 126–147, Heidelberg, Germany, 2001. Springer-Verlag.
- [5] J. Fisher and A. Craig. Developing business community portals for SMEs - issues of design, development and sustainability. *Electronic Markets*, 15(2):136–145, May 2005.
- [6] N. Mavetera and A. Kadyamatimba. A comprehensive agent-mediated e-market framework. In *Proceedings of the 5th International Conference on Electronic Commerce (ICEC'03)*, pages 158–164, New York, NY, USA, 2003. ACM Press.
- [7] I. Seidel and H. Berger. Integrating electronic institutions with 3d virtual worlds. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*, pages 481–484, Washington, DC, USA, 2007. IEEE Computer Society.
- [8] G. Smith, M. L. Maher, and J. S. Gero. Designing 3d virtual worlds as a society of agents. In *Proceedings of the 10th International Conference on Computer Aided Architectural Design Futures (CAADFutures'03)*, pages 105–114, Dordrecht, Netherlands, 2003. Kluwer Academic Publishers.
- [9] D. Traum and J. Rickel. Embodied agents for multi-party dialogue in immersive virtual worlds. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 766–773, New York, NY, USA, 2002. ACM Press.
- [10] S. Wang, S. Zheng, L. Xu, D. Li, and H. Meng. A literature review of electronic marketplace research: Themes, theories and an integrative framework. *Information Systems Frontiers*, 10(5):555–571, 2008.

# Semantic Service Matchmaking in the ATM Domain Considering Infrastructure Capability Constraints

Thomas Moser, Richard Mordinyi, Wikan Danar Sunindyo, Stefan Biffel

*Institute of Software Technology and Interactive Systems, Vienna University of Technology*

*Favoritenstrasse 9-11/188, Vienna, Austria*

*{thomas.moser, richard.mordinyi, wikan.sunindyo, stefan.biffel}@tuwien.ac.at*

**Abstract**—In a service-oriented environment business processes (BPs) flexibly build on software services (SSs) provided by systems in a network. A key design challenge is the semantic matchmaking of BPs and SSs in two steps: 1. Find for one BP the SSs that meet or exceed the BP requirements; 2. Find for all BPs the SSs that can be implemented within the capability constraints of the underlying network, which poses a major problem since even for small scenarios the solution space is typically very large. In this paper we analyze requirements from mission-critical BPs in the Air Traffic Management (ATM) domain and introduce an approach for semi-automatic semantic matchmaking for SSs, the “System-Wide Information Sharing” (SWIS) BP integration framework. A tool-supported semantic matchmaking process like SWIS can provide system designers and integrators with a set of promising SSs candidates and therefore strongly reduces the human matching effort by focusing on a much smaller space of matchmaking candidates. We evaluate the feasibility of the SWIS approach in an industry use case from the ATM domain.

## I. INTRODUCTION

Safety-critical systems and business processes, e.g., in the Air Traffic Management (ATM) domain, have to become more flexible to implement changes due to new business environments (e.g., mergers and acquisitions), new standards and regulations. A promising approach follows the service-oriented architecture (SOA) paradigm that builds flexible new systems for business processes (BPs) based on a set of software services (SSs) provided by system nodes in a network. A key design challenge is the matchmaking of BPs and SSs, i.e., finding the SSs that a) best meet the requirements of the BPs under consideration and b) can be implemented with the available network capabilities. The solution space is typically large even for small problems and a general semantic solution to enable comprehensive tool support seems infeasible.

To provide a SOA solution for a set of BPs, meaning to identify suitable SSs for BPs, designers and system integrators need to overcome 3 integration challenges that build on each other:

1. *Technical integration* connects networked systems that use heterogeneous technologies, i.e., different protocols, operational platforms, etc. Current technical integration approaches like Enterprise Service Bus (ESB) [2] or Service Oriented Architecture (SOA) [14] need manual configuration on the technical detail level and tool support is typically focused on a single technology or vendor.

2. *Semantic integration* translates data content and format between systems that use heterogeneous semantics, i.e., different terminologies for service names, data formats, etc. For

semantic integration, there is no standard or framework available, making the semantic transformations between multiple services inefficient and expensive.

3. *Business process support* builds on technically and semantically integrated systems that provide SSs the BP needs to fulfil its goal. The system integrator has to select SSs that really match the requirements of the BP, and check whether the infrastructure capabilities can support the communication requirements of the chosen solution.

Large BP and SS integration networks consist of hundreds of integration nodes; changes of SS properties and network capabilities make the correct and efficient identification of feasible BP and SS pairs a recurring complex and error-prone task. Current service matchmaking approaches focus on either technical or semantic integration issues [20], while business process support is, to our knowledge, missing. Tool support for matchmaking of BPs and SSs need to make the requirements of BPs and SSs as well as the capabilities of SSs and the underlying infrastructure understandable for machines.

In previous work, we introduced a systems integration approach, the “system-wide information sharing” (SWIS) approach. The SWIS framework explicitly models the semantics of integration requirements and capabilities in machine-understandable form (semantic integration) [17]; and the connectors and transformations between heterogeneous legacy systems (technical integration) to simplify systems integration (business process support) [16].

In this paper, we describe the semantic matchmaking of BPs and SSs and the optimization of the integration solution with respect to available network capabilities. Semantic matchmaking uses the machine-understandable SWIS models to describe BP and SS requirements and SS and network capabilities to derive 2 results: 1. Provide sets of possible SSs for each BP; 2. Optimize the set of selected SSs with multiple objectives (e.g., costs, delay) while observing the capabilities of the underlying network infrastructure, a variation of the knapsack problem [11]. We evaluate the feasibility of the SWIS approach in a use case from the ATM domain.

The remainder of this paper is structured as follows: Section 2 summarizes related work, Section 3 motivates the research issues, while Section 4 describes the use case. Section 5 elaborates the semantic service matchmaking approach and the optimization of the integration solution. Section 6 evaluates the approach and Section 7 discusses the results with regard to the research issues. Finally, Section 8 concludes and identifies further work.

## II. RELATED WORK

This section summarizes related work on technical integration, semantic integration with semantic web services, and service matchmaking with multi-objective optimization.

### A. Technical Integration

Technical system integration is the task to combine networked systems that use heterogeneous technologies to appear as one big system. There are several levels at which system integration could be performed [1], but there is so far no standardized integration process that explains how to integrate systems in general.

The need for integration over heterogeneous middleware technologies with different APIs, transportation capabilities, or network architecture styles implies either solutions like ESB [2] and SOA [14] or the development of static and therefore inflexible wrappers between each combination of middleware technologies, and thus increases the complexity of communication.

### B. Semantic Integration with Semantic Web Services

Semantic integration is solving problems originating from the intent to share data across disparate and semantically heterogeneous data sources [6]. These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modeling of complex relations in different sources [18]. One of the most important and most actively studied problems in semantic integration is establishing semantic correspondences (or mappings) between vocabularies of different data sources. [3]

The use of ontologies as a solution option to semantic integration and interoperability problems has been studied over the last 10 years. Ontologies promise to provide machine-understandable representation of knowledge, while allowing the mapping between certain facts as well as the derivation of new facts using reasoning approaches based on the modeled knowledge [21]. In a general domain, semantic integration has shown to be very hard if not unsolvable. However, in a specialized domain, like the ATM domain, semantic integration seems doable. Noy [18] identified three major dimensions of the application of ontologies for supporting semantic integration: the task of finding mappings (semi-)automatically, the declarative formal representation of these mappings, and reasoning using these mappings.

In SOA the promise of Web Services and the need for widely accepted standards enabling them are by now well recognized [7]. At the same time, recognition is growing of the need for richer semantic specifications of Web Services, so as to enable fuller, more flexible automation of service provision and use, support the construction of more powerful tools and methodologies, and promote the use of semantically well-founded reasoning about services. Furthermore, richer semantics can help to provide fuller automation of activities as verification, simulation, configuration, supply chain management, contracting, and negotiation of services. [12]

To meet this need, researchers have been developing languages, architectures and related approaches for so called Se-

mantic Web services [13]. The Ontology Web Language for Services (OWL-S), which seeks to provide the building blocks for encoding rich semantic service descriptions in a way that builds naturally upon the Web Ontology Language (OWL), supplies Web Service providers with a core set of markup language constructs for describing the properties and capabilities of their Web Services in unambiguous, computer-interpretable form [4]. WSDL-S [15] is another approach for annotating current Web Service standards with semantic descriptions. The Web Service Modeling Ontology (WSMO) [10] is a framework for Semantic Web Services which defines a rich conceptual model for the development and the description of Web Services based on two main requirements: maximal decoupling and strong mediation.

All three approaches, OWL-S, WSDL-S and WSMO, provide mechanism for semantically describing Web Services, with the major goal of allowing generic description of service functionality as well adding semantics to general service descriptions like provided/consumed messages or service bindings. This ambitious goal seems very useful for generic service descriptions; however its usage is limited in specific domains like in the ATM domain, since too specific features would complicate a generic approach too much. Therefore, we defined our own ontology-based architecture for describing the properties and features of the ATM services [17].

### C. Service Matchmaking Approaches

Semantic matchmaking can be seen as major feature of semantic integration which supports designers and system integrators by providing sets of possible integration partners regarding both structural and semantic attributes. However, the relevant semantic concepts are hard to define unambiguously for general domains, thus the focus on a well-defined domain like ATM provides semantic clarity.

Kolovski et al. [8] provide a mapping of WS-Policy to OWL. WS-Policy provides a general purpose model and syntax to describe the policies of a Web service. It specifies a base set of constructs that can be used and extended by other Web service specifications to describe a broad range of service requirements and capabilities. The main advantage of representing Web Service policies using OWL is that OWL is much more expressive than WS-Policy and thus provides a framework for exploring richer policy languages. Verma et al. [20] present an approach for matching the non-functional properties of Web Services represented using WS-Policy. Oldham et al. [19] present a framework for the matching of providers and consumers based on WS-Agreements. The WS-Agreement specification defines a language and protocol for capturing relationships with agreements between two parties.

Both WS-Policy and WS-Agreement define a generic framework for the representation of standard Web Service policies, however both frameworks seem too generic to be effectively used in a concrete scenario from a specialized domain like the ATM domain is. Therefore, we used the concept of describing Service policies using a knowledge representation language like OWL, but defined our own extendable policy representation language which is better suitable for the ATM domain [17].



### III. RESEARCH ISSUES

Recent projects with industry partners from the ATM domain raised the need for semi-automated BP integration support in technology-driven integration environments. Recently, we developed a data-driven approach [16] that explicitly models the semantics of the problem space, i.e., BP integration requirements and network infrastructure capabilities [17]; the solution space, i.e., the connectors, and data transformations between SSs. Finally, we provide a process to bridge problem and solution spaces, i.e., identify feasible BP and SSs pairs while fulfilling business requirements and optimizing the chosen integration solution according to multiple objectives.

Figure 1 provides an overview on the integration layers, data flows between the integration layers, and the steps of the semantic service matchmaking process: SM1: For each BP, identify the suitable SSs sets, which fulfil all BP service and data requirements. From these possible BP and SSs sets, the system integrators choose the most promising sets, the so-called collaboration sets. SM2: The selected collaboration sets are then optimized regarding the original infrastructure requirements of both the business BPs and the SSs, as well as the available limited capabilities of the infrastructure's nodes and links. The outcome of SM2 is an optimized configuration of the integration solution, consisting of the selected collaboration sets as well as their grounding to the underlying integration network infrastructure.

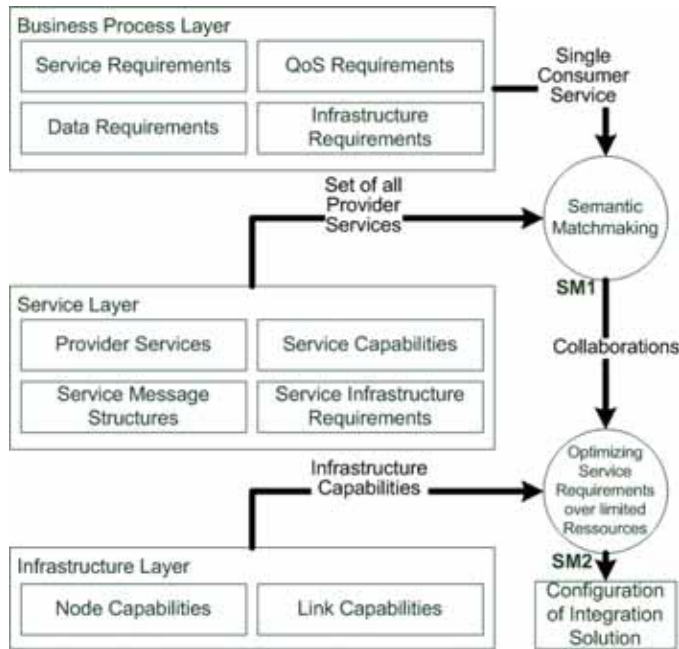


Figure 1: Semantic Service Matchmaking Process Steps.

Based on this, we derive the following research issues:

**RI-1: Semantic Matchmaking of SS candidates for one BP (SM1).** Provide machine-understandable descriptions for BP and SSs requirements as well as SS and network capabilities to provide tool support for SM1 to make the search space reduction effective (low number of false negatives and false

positives) and efficient (less human effort required) compared to the current human-based approach.

**RI-2: Resource Feasibility Check and Optimization for all Collaborations (SM2).** Provide a framework to enable a) checking the validity of a set of BPs and SSs with the infrastructure capability constraints and b) ranking valid solutions by multiple optimization criteria like network cost and service delay.

### IV. ATM SCENARIO DESCRIPTION

This section describes the integration scenario from the ATM domain used throughout this paper. The ATM use case (Figure 1) represents information that is typically extracted from customers/participants in workshops on requirements and capabilities elicitation for information systems in the aviation domain. In safety-critical domains like ATM BP integration solutions have to pass certifications before deployment, which typical dynamic SOA solutions [2, 14] cannot fulfil regarding the current rigid integration network in the ATM domain designed to guarantee integration requirements even in case of partial failure.

In the ATM domain semantic matchmaking is an effort for scarce human experts who have to cope with a huge search space and often miss better solutions due to their simple heuristic search strategies. Tool-supported semantic matchmaking provides designers and system integrators with a set of promising integration partner candidates and therefore strongly reduces the human matching effort by focusing on a much smaller space of feasible matchmaking candidates that can be rated according to relevant optimization criteria.

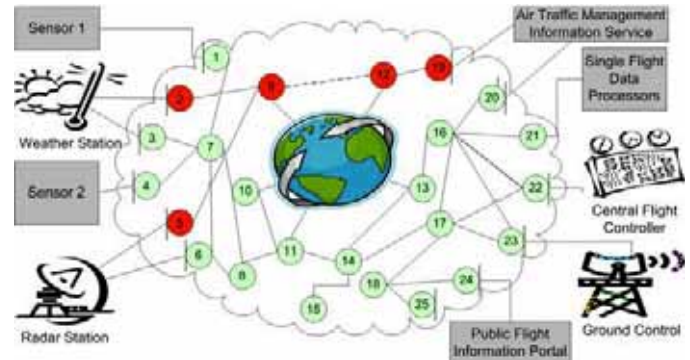


Figure 2: A Typical ATM Domain Integration Network.

As shown in Figure 2, the integration network consists of business services connected to integration network nodes. Between these nodes, there may exist different kinds of network links using different transmission technologies (e.g., radio or wired transmission) as well as different middleware technologies for communication purposes. The capabilities of nodes and links, like throughput, availability, reliability, or security are explicitly modelled in order to be capable of selecting suitable communication paths for particular service requirements, e.g., the communication link between the red ATMIS Node and the red Node 12 represents a reliable and secured communication path which may be requested by e.g., the ATMIS business service.

## V. SEMANTIC SERVICE MATCHMAKING

This section describes the semantic service matchmaking approach as well as the multi-objective optimization of the chosen integration services candidates.

### A. Identification of Possible Collaboration Candidate Sets

The identification of possible collaboration candidate sets is implemented as a heuristic algorithm. Step by step, the possible collaboration candidate sets are reduced by applying the rules described to the possible collaboration candidate sets. The heuristic rules that are applied during the source/sink matching are described in the following paragraphs.

**Message mapping.** During the description of the SS messages, each SS message segment was mapped to a domain concept, which has been specified in the common domain ontology. Therefore, for all segments of the message required by a certain BP, it is searched for messages of the SSs that contain segments, which are mapped to the same domain concept, and if possible, to the same message format.

**Service Policies.** In addition, SSs can define requirements (policies) regarding preferred or unwanted SS partners, as well as other non-functional requirements, e.g., QoS requirements regarding the underlying integration network. A policy is a restriction or a condition for a single collaboration or a set of collaborations, in order to allow the communication via the underlying integration network. In SWIS-based applications, there are two kinds of policies. On the one hand, there are policies which are valid for all collaborations. They specify global conditions that need to be fulfilled by all collaborations, e.g., a maximum time for the delivery of messages. On the other hand, there are policies which are required only for a specific subset of collaborations. These policies specify conditions that need to be fulfilled by the collaborations containing particular SSs, e.g., the communication has to use only secure links, or only a specified set of other SSs is allowed to participate in the collaboration. The SS policies that regard other SSs are evaluated by checking whether the attributes and tags of every SS of the particular collaboration candidate meet the service policies defined by the BP.

**Format Translation.** If a message segment is mapped to the same domain concept as the required message segment, but the formats of the two segments differ, check whether there is a converter defined for the two formats. A converter is used to convert the format of message segments from one basic data type to a different one. An explicit identifier is defined to allow the search for the converter at runtime (e.g., by using Java Reflection).

**External Service Transformation.** If the message segments differ in the domain concept they are mapped to, check if a service exists that consumes a segment mapped to the same domain concept as the segment of the message of the SS and provides a message with a segment mapped to the same domain concept of the segment of the message of the BP.

**Route Deduction.** As last rule it is checked whether there is an existing route between the nodes connecting the SSs and the node connecting the BP.

If all the rules mentioned above are successfully applied to a set of one or more SSs and a BP, then the particular set is accepted as collaboration candidate. If any of the rules cannot be met, the particular set is discarded as collaboration candidate.

### B. Validity-Check and Optimization of Collaborations

Once all collaborations have been specified a Scenario is derived. A Scenario contains beside all collaborations a specification detailing how to configure the network infrastructure, so that the integration solution is optimized according to the given objectives. In the following the process steps needed to optimize the scenario is explained.

**Preliminary Checks.** The process step checks whether there is at least one single network route for each collaboration satisfying all global and collaboration specific policies. If this step cannot be completely satisfied the process raises an exception. The system integrator either updates or removes the collaborations which cannot be mapped to a network route, and restart the process step, or adapts the semantic infrastructure model, by adding additional nodes and links.

**Route Derivation.** Once it has been verified that each collaboration can be mapped to at least one route in the network, the process step derives every possible route for each collaboration. The only restrictions are that no node is allowed to appear twice within the same route and all policies have to be satisfied. The valid ones are retained; the ones violating the restrictions are removed. At the end of this process step, each collaboration will have either a single route or a set of valid routes to choose from.

**Creating Scenarios.** The processing step combines each route of each collaboration with each other. This means that a scenario consists of a set of collaborations where each collaboration represents exactly one route. The more scenarios are created, the higher the probability to find a scenario that is well suited for achieving the stated optimization objectives.

**Evaluation.** The process iterates through all scenarios and calculates their fitness according to the optimization objectives. The fitness of a scenario is the fitness of all its containing collaborations, and represents the real values (e.g. the time a message needs and the costs along the chosen route) of the objectives. The fitness represents the trade-off of the configuration, the routes of each collaboration predetermine. The set of fitness values is then analyzed according to the Pareto Front approach [5]. The Pareto Front contains either a single Scenario or a set of Scenarios. In the latter case there may be several “nearly equivalent” configurations as integration solutions. Thus, the system integrator has to decide which one to pick for practical deployment.

**Multi-Objective Optimization.** We have accomplished the process of optimizing collaborations by implementing a Java version of the mPOEMS approach into the SWIS framework. mPoems is an evolutionary algorithm using the concept of dominance for multi-objective optimization. The results and explanations of the approach can be found at [9].

## VI. EVALUATION

In this section, we evaluate the SWIS framework using a clear and comprehensible example to show the principles of our approach.

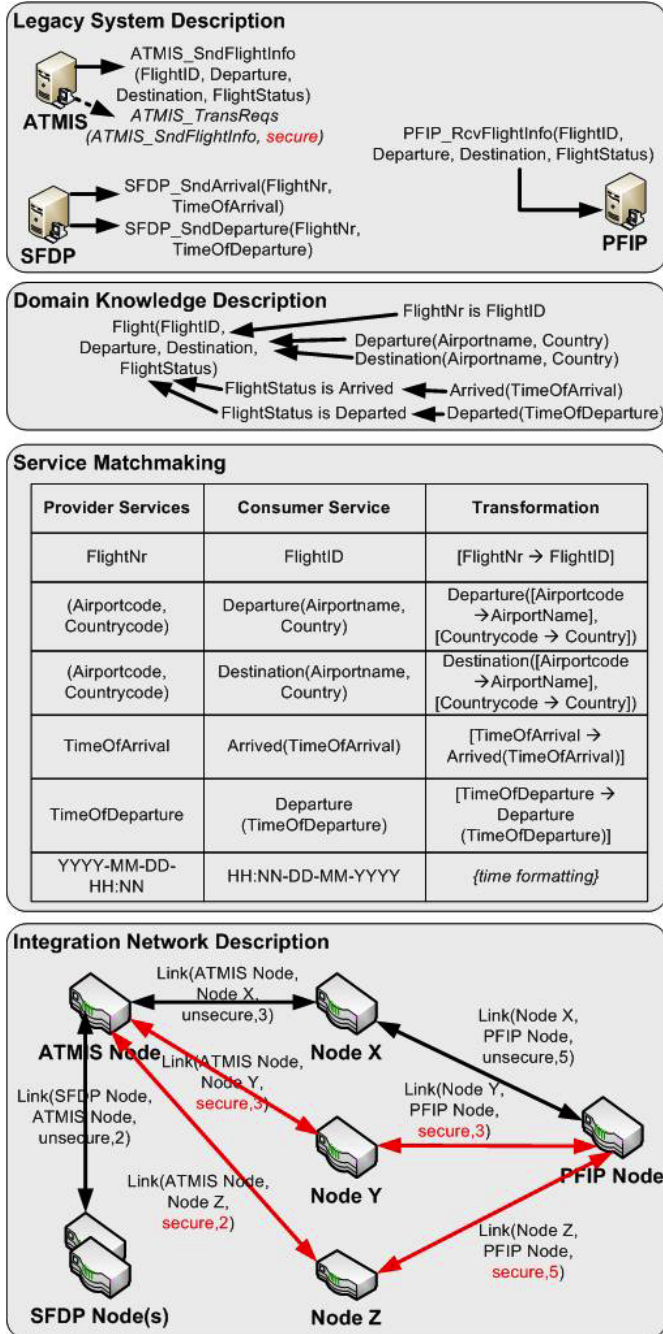


Figure 3: Service Matchmaking Example.

An example for semantic service matchmaking in the SWIS framework is shown in Figure 3. There are three services of provided by legacy systems, two provider services (*ATMIS* and *SFDP*) and one consumer service (*PFIP*). The consumer service needs information that can be obtained from the provider services, i.e. *FlightID*, *Departure*, *Destination* and *FlightStatus*. This needed information is provided separately

by the two provider services, so the system has to find the suitable information that match with the consumer service's needs. Additionally, the service *ATMIS\_TransReqs* defines a policy for the underlying integration network, stating that only secure network links may be used for the communication.

From the domain knowledge description, we know that *Flight ID* is a synonym for *Flight Number*, that *Departure* and *Arrival* are combinations of the airport code and country code of departure/arrival, and that the *FlightStatus* arrived or departed, can be derived by checking the occurrence of either *TimeOfArrival* or *TimeOfDeparture*.

Next, we calculate the network resources needed for sending messages from the *SFDP* Node to the *PFIP* Node with less capacity. From the integration network description, we can see several nodes connected by links. Each link contains information regarding source node and target node, support for secure transmissions and the transmission delay. The communication between *ATMIS* to *PFIP* needs to be done using secure connections only. There are two possible connections, either via *Node Y* or via *Node Z*. The system will choose connection via *Node Y* because it has less delay (6) than connection via *Node Z* (7).

## VII. DISCUSSION

The example shows that even for small problems the solution space is typically large. However, large BP and SS integration networks consist of hundreds of integration nodes; and changes of SS properties and network capabilities make the correct and efficient identification of feasible BP and SS pairs a recurring complex and error-prone task. By providing only sets of feasible/promising service provider and consumer candidates, semantic matchmaking supports designers and system integrators by providing sets of possible integration partners regarding both structural and semantic attributes. However, the relevant semantic concepts are hard to define unambiguously for general domains, thus the focus on a well-defined domain like ATM provides semantic clarity.

We used the concept of describing Service policies using a knowledge representation language like OWL, but defined our own extendable policy representation language which is better suitable for the ATM domain. We do not use standardized Web Service description frameworks because, since the strengths of Web Service description frameworks lies in the generality of the approach, however their weakness is that it may become complicated to describe domain-specific issues. For specific domains, it may be useful to use the principles of web service descriptions but tailor them to the domain. Additionally, we defined our own ontology-based architecture for describing the properties and features of the ATM services.

We have developed a data-driven approach [16] that explicitly models the semantics of the problem space, i.e., BP integration requirements and network infrastructure capabilities [17]; the solution space, i.e., the connectors, and data transformations between SSs. In this paper, we described a process to bridge problem and solution spaces, i.e., identify feasible BP and SSs pairs while fulfilling business requirements and optimizing the chosen integration solution according to multi-

ple objectives. In order to evaluate the proposed process, we have derived two major research issues that will be discussed in the following paragraphs.

**Semantic Matchmaking of SS candidates for one BP.** Current service matchmaking approaches focus on either technical or semantic integration issues [20], while business process support is, to our knowledge, missing. In the SWIS framework, we presented a combined service matchmaking approach that performs matching based on the data of the services and available service policies regarding other services. The SWIS framework's semantic service matchmaking enables an effective search space reduction and poses lower risk and effort compared to the current human-based approaches.

**Resource Feasibility Check and Optimization for all Collaborations.** The optimization process steps allow using existing resources efficiently. Out of all possible collaborations for a single business process which are creatable by means of the proposed semantic matchmaking approach, only those are desirable to be deployed in the integration solution which fulfill certain criteria. Those criteria are set up by the integration expert so that existing collaborations use the underlying integration network infrastructure with its limited resources as efficient as possible.

## VIII. CONCLUSION AND FURTHER WORK

In this paper we presented an approach for semi-automatic semantic matchmaking for software services (SSs), the "System-Wide Information Sharing" (SWIS) Business Process (BP) integration framework. The SWIS BP integration frameworks uses the machine-understandable SWIS models to describe BP and SS requirements as well as SS and network capabilities to provide sets of possible SSs for each BP. Out of these possible sets, the system integrators choose the wanted sets. These wanted sets are then optimized with multiple objectives (e.g., costs, delay) while observing the capabilities of the underlying network infrastructure.

We evaluated the feasibility of the SWIS approach in an industry use case from the ATM domain. The example shows that even for small problems the solution space is typically large, and even bigger for large BP and SS integration networks consisting of hundreds of integration nodes. A tool-supported semantic matchmaking process like SWIS can provide system designers and integrators with a set of promising SSs candidates and therefore strongly reduces the human matching effort by focusing on a much smaller space of matchmaking candidates.

**Further Work.** Further work will include a detailed description of the semantic design to translate between matched services and an evaluation measuring the effectiveness and efficiency of deriving the semantic transformation with tool-support compared to a manual approach.

## ACKNOWLEDGMENT

The authors would like to acknowledge all project members of the SWIS (System-Wide Information Sharing) project performed from 2006-2008 at Vienna University of Technology together with Frequentis AG and Austro Control GmbH.

## REFERENCES

- [1] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema, "Developing Applications Using Model-Driven Design Environments," *COMPUTER*, 2006, pp. 33-40.
- [2] D.A. Chappel, *Enterprise Service Bus*, O'Reilly Media, 2004.
- [3] A. Doan, N.F. Noy, and A.Y. Halevy, "Introduction to the special issue on semantic integration," *SIGMOD Rec.*, vol. 33, no. 4, 2004, pp. 11-13.
- [4] J. Dong, Y. Sun, and S. Yang, "OWL-S Ontology Framework Extension for Dynamic Web Service Composition," *18th Intl Conf on SE & Knowledge Engineering (SEKE'2006)*, 2006, pp. 544-549.
- [5] M. Ehrgott, *Multicriteria Optimization*, Springer, 2005.
- [6] A. Halevy, "Why your data won't mix," *Queue*, vol. 3, no. 8, 2005, pp. 50-58.
- [7] S. Herr, K. Läufer, J. Shafae, G.K. Thiruvathukal, and G. Wirtz, "Combining SOA and BPM Technologies for Cross-System Process Automation," *20th Intl Conf on SE & Knowledge Engineering (SEKE'2008)*, 2008, pp. 339-344.
- [8] V. Kolovski, B. Parsia, Y. Katz, and J. Hendler, "Representing Web Service Policies in OWL-DL," *4th International Semantic Web Conference (ISWC 2005)*, Springer, 2005, pp. 461-475.
- [9] J. Kubalik, R. Mordinyi, and S. Biffl, "Multiobjective Prototype Optimization with Evolved Improvement Steps," *Evolutionary Computation in Combinatorial Optimization*, 2008.
- [10] H. Lausen, A. Polleres, and D. Roman, "Web Service Modeling Ontology (WSMO)," *W3C Member Submission*, vol. 3, 2005.
- [11] S. Martello, and P. Toth, *Knapsack problems: algorithms and computer implementations*, John Wiley & Sons, 1990.
- [12] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, and M. Solanki, "Bringing Semantics to Web Services: The OWL-S Approach," *First International Workshop on Semantic Web Services and Web Process Composition*, Springer, 2005, pp. 26-42.
- [13] S.A. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services," *IEEE INTEL. SYSTEMS*, vol. 16, no. 2, 2001, pp. 46-53.
- [14] P.P. Mike, and H. Willem-Jan, "Service oriented architectures: approaches, technologies and research issues," *The VLDB Journal*, vol. 16, no. 3, 2007, pp. 389-415.
- [15] J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam, "WSDL-S: Adding Semantics to WSDL-White Paper," 2004.
- [16] T. Moser, R. Mordinyi, A. Mikula, and S. Biffl, "Efficient System Integration Using Semantic Requirements and Capability Models: An approach for integrating heterogeneous Business Services," *11th International Conference on Enterprise Information Systems (ICEIS 2009)*, 2009, accepted for publication.
- [17] T. Moser, R. Mordinyi, A. Mikula, and S. Biffl, "Making Expert Knowledge Explicit to Facilitate Tool Support for Integrating Complex Information Systems in the ATM Domain," *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2009)*, 2009, accepted for publication.
- [18] N.F. Noy, "Semantic integration: a survey of ontology-based approaches," *SIGMOD Rec.*, vol. 33, no. 4, 2004, pp. 65-70.
- [19] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour, "Semantic WS-agreement partner selection," *15th International World Wide Web Conference, ACM*, 2006, pp. 697-706.
- [20] K. Verma, R. Akkiraju, and R. Goodwin, "Semantic Matching of Web Service Policies," *2nd International Workshop on Semantic and Dynamic Web Process (SDWP 2005)*, 2005.
- [21] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner, "Ontology-based integration of information—a survey of existing approaches," *Workshop on Ontologies and Information Sharing (IJCAI-01)*, 2001, pp. 108-117.

# Ontology Mapping Representations: a Pragmatic Evaluation

Hendrik Thomas, Declan O’Sullivan and Rob Brennan

Knowledge & Data Engineering Group, School of Computer Science and Statistics, Trinity College Dublin, Ireland  
Email: {Hendrik.Thomas, Declan.OSullivan, Rob.Brennan}@cs.tcd.ie

*Abstract—A common approach to mitigate the effects of ontology heterogeneity is to discover and express the specific correspondences (mappings) between different ontologies. An open research question is: how should such ontology mappings be represented? In recent years several proposals for an ontology mapping representation have been published, but as yet no format is officially standardised or generally accepted in the community. In this paper we will present the results of a systematic analysis of ontology mapping representations to provide a pragmatic state of the art overview of their characteristics. In particular we are interested how current ontology mapping representations can support the management of ontology mappings (sharing, re-use, alteration) as well as how suitable they are for different mapping tasks.*

## I. INTRODUCTION

Ontologies are an important component for implementation of the semantic web vision [1]. The promise of ontologies is to enable the sharing of a common understanding of a domain that can be flexibly communicated between users and applications [2,3]. However, the actual conceptualisation of a domain and the subsequent explication in an ontology language is a very heterogeneous process [4]. For example conceptual heterogeneity arises due to the natural human diversity involved in modelling a domain [5], e.g. two ontologies could differ because they provide a more (or less) detailed description or could reflect different viewpoints of the same domain. These different levels of heterogeneities [6] are major obstacles to the promise of interoperability based on ontologies [7].

A common approach to mitigate the effect of heterogeneity is to discover the specific correspondences between the different ontologies and to document these correspondences using an appropriate ontology mapping expression [8,9]. We define ontology mapping as the task of relating the vocabulary of two ontologies sharing a domain in such a way that the structure of ontological signatures and their intended interpretations are respected [10].

One aspect, which is still open to discussion, is: how should ontology mappings be represented [6,8,11]? In this paper we define an ontology mapping representation as an explicit specification of the correspondence between ontologies to improve their interoperability. In recent years several proposals and recommendations for such an ontology mapping representation have been published but as yet no representation specific format is standardised or even generally accepted in the semantic web community [8,12]. Thus an ontology engineer, when confronted with the need to merge or align multiple ontologies has a choice between multiple currently available ontology mapping representations, each with their individual strengths and weaknesses for a specific mapping task.

Publications focusing on representations of ontology mapping are relatively rare compared to the huge number focusing on other related questions for matching and mapping, e.g. matching algorithms to identify mapping candidates [13]. However, some previous studies on ontology matching and mapping systems provide some insight [10,11,12]. Most of these previous evaluations focus primarily on the technical capabilities of matching and mapping tools [12,14] and less on applicability of mappings representations [6]. In addition, only sparse information has been published on the support of reusability and management of mappings, e.g. specification of supporting meta-data [11]. Finally, the evaluation processes and criteria sets used in previous work have been quite heterogeneous which makes it difficult to identify trends and improvements over time. In summary, a detailed evaluation framework as well as a comprehensive and up-to-date evaluation focusing on the capabilities of current ontology mapping representations is currently missing.

In this paper we present the results of a systematic analysis of ontology mapping representations to provide a pragmatic state of the art overview of their characteristics. In particular we are interested how the ontology mapping representations can support the management of ontology mappings (sharing, re-use, alteration) as well as how suitable they are for different mapping tasks. The results of this evaluation will be of interest for understanding ontology mapping interoperability issues and will also support ontology engineers in choosing the most suitable mapping representation for their application.

## II. EVALUATION FRAMEWORK

### A. Methodology

To derive metrics for an evaluation of mapping representations we apply the Goal Question Metric (GQM) method as a tried and tested method for a structured and replicable evaluation of software products [15]. GQM provides a hierarchical structured procedure starting with goals for each relevant evaluation dimension [15]. Each goal is refined into several questions to break down the issue to characterize the object of measurement. Each question is then refined into metrics (objective, subjective) in order to answer it in a quantitative way. The result of the application of the GQM method is a replicable and detailed specification of a measurement system targeting a particular set of relevant issues [15]. In the following subsections we give a brief introduction to our evaluation framework for ontology mapping representations derived using the GTM method. The framework reflects our evaluation focus on management and applicability of ontology mapping representations. A detailed description of the framework can be found in [16].

### B. Evaluation Goals, Questions and Metrics

Turning to the literature of ontology mappings it can be observed that instances of mapping relations can be quite heterogeneous, ranging from simple equivalence relations or mathematical conversions to complex structural mappings [5,11,12]. Therefore one of the fundamental goals of ontology mapping representations is (G1) *the ability to express a mapping relation*. Relevant for this goal are the ontology elements which can be addressed as well as the different kinds of mapping relation types which are supported. In addition we need to consider the supported operators and functions for the expression of conversion and structural mappings. Table I gives an overview of all deduced criteria for this goal [16].

TABLE I GOAL 1: ABILITY TO EXPRESS A MAPPING RELATION

Criteria	Type	Examples
<i>Question 1: Which kind of ontology elements can be addressed?</i>		
Single ontology element	yes/no	OWL class, property
Ontology fragment	yes/no	SELECT ?x WHERE {?x ?y ?z}
Ontology as a whole	yes/no	http://kdeg.org/nembes.owl
<i>Question 2: Which relations types are predefined?</i>		
Count of predefined types	0..X	3
List of predefined types	list	equivalence, subsumption
Extensibility	yes/no	add a "neighbor" relation
<i>Question 3: Which function for conversion mappings are supported?</i>		
Numerical function	yes/no	add, subtract, multiply
String functions	yes/no	delete leading white spaces
Date functions	yes/no	2006/12/31 to 31/12/2006
<i>Question 4: Which function for structural mappings are supported?</i>		
Add / remove classes	yes/no	remove class town
Add / remove instances	yes/no	add instance Dublin
Add remove relation	yes/no	add Dublin is-part-of Ireland
Add remove attributes	yes/no	remove a variant name

The second aspect is that (G2) *ontology mapping representation should be computationally efficient to process* [7,11] in order to support the pragmatic concerns of implementing ontology interoperability solutions. Table II gives an overview of all deduced criteria for this goal.

TABLE II GOAL 2: COMPUTATIONALLY EFFICIENT TO PROCESS

Criteria	Type	Examples
<i>Question 1: How is the compatibility of the representation?</i>		
Implementation independent	yes/no	MAFRA format
Syntax	yes/no	XML, RDF, OWL
<i>Question 1: Which tool support the mapping representation?</i>		
Creation & editing tools	List	Ontology Alignment API
Sharing tools	List	CVS
Management tools	List	COMA++
Mapping visualization tools	List	MAFRA

Besides these aspects we need to consider that the construction of a specific ontology mapping can be complex and time-consuming [12]. Instead of creating the same or similar mappings repeatedly it is important to have a goal (G3) *to enable sharing and reuse of existing mappings* to reduce the effort involved in the creation of mappings [7,8]. To decide if an ontology mapping can be reused, it is essential to understand how the mapping was created. An analysis of the life cycle of an ontology mapping [7] is helpful for identification of relevant decisions and information, e.g. which matching algorithms have been used [8,18]. Meta-data documenting this lifecycle is needed to facilitate sharing and reuse of mappings. An ontology mapping representation should provide suitable placeholders to make this information retrievable in a structured and predictable way. Previously we have defined a mapping lifecycle [8] and based on that we identified meta-data to document the source and target ontologies, the matching phase to identify mapping candidates, the map-

ping phase as well as the management phase. Table III gives an overview of all deduced criteria for this goal.

TABLE III GOAL 3: SHARING & REUSING OF EXISTING MAPPINGS

Criteria	Type	Examples
<i>Question 1: How are the sources and target ontologies documented?</i>		
Ontology identifiers	yes/no	string based matcher
Version information	yes/no	ontology version 1.5.4.
Ontology format(s)	yes/no	OWL lite, RDF(s)
Canonical format	yes/no	XML schema [8]
Terms used	yes/no	link to relevant thesauri
Ontology measures	yes/no	count of classes
<i>Question 2: How is the matching phase documented?</i>		
Matching policies applied	yes/no	policy of organization A
Matching creation type	yes/no	automated or manual
Info on manual matching	yes/no	link to documentation
Identify used matcher	yes/no	model based matcher
Matcher configuration	yes/no	parameter
Matcher type	yes/no	linguistic based matcher
<i>Question 2: How is the mapping phase documented?</i>		
Matching policies applied	yes/no	policy of organization A
Used pre-validated mappings	yes/no	A;creator = B;author
Mapping context	yes/no	specification of use-cases
Confidence level	yes/no	5 of 10
Mapping strategy	yes/no	OISIN framework [8]
<i>Question 3: How is the management phase documented?</i>		
Distribution system	yes/no	peer-to-peer network
Version information	yes/no	map version 1.2.3
Format information	yes/no	INRIA 1.0
Conflict/consistency check	yes/no	conflict mapA vs. mapB
Author information	yes/no	Hendrik Thomas
Date of creation	yes/no	19.12.2008 17:00
Authority for changes	yes/no	see http://onto.authority.ie
Dependencies	yes/no	mapping A depends on B
Change propagation	yes/no	newsgroups announcement
<i>Question 4: How is the interpretation of the meta-data supported?</i>		
URI to identify entities	yes/no	http://cs.tcd.ie/onto/fname
Human-readable labels	yes/no	first Name
Documentation of meta data	List	source code, publications
Documentation URI	yes/no	http://cs.tcd.ie/onto/docu
Ontology identifiers	yes/no	URL of ontology source

Another relevant issue for this evaluation framework is: which ontology mapping representations should be included in the evaluation? Currently there are several non-ontology based (e.g. Text, XML) and ontology based (e.g. RDF, OWL [1]) languages used to express mappings [7]. The problem is that there is no consistent usage of these languages or formats. In fact, many mapping tools use the same languages to express mapping results (e.g. RDF is very common) but in different ways and as a consequence they support different functions and operators to express mappings [7,8]. From a pragmatic point of view it is therefore not enough to evaluate a representation language like OWL in isolation. In fact, it is important to understand which specific instances of ontology mapping representations are supported by the individual mapping tools.

### III. EVALUATION RESULTS

In this evaluation we analyzed 13 different mapping and matching applications (see appendix for a complete list). The selection include historically relevant and established tools but also examples of up-to-date matching applications [24]. For each of the 22 supported ontology mapping representation instances, 31 different evaluation parameters were determined. The evaluation was conducted in early 2009 by the authors in the Knowledge and Data Engineering Group, Trinity College (Dublin). The complete evaluation results are available online at: [https://www.cs.tcd.ie/~thomash/mapping\\_eva/results.php](https://www.cs.tcd.ie/~thomash/mapping_eva/results.php)

### A. Results for G1 Ability to Express a Mapping Relation

The first aspect we analyzed in our evaluation was the expressiveness of the application in terms of which operators and functions are supported to express mappings. We noticed that all analysed tools are limited to addressing individual ontology elements as subjects of mappings (Q1). Thus none of the evaluated applications is able to address ontology fragments which is quite odd because mappings of complex statements need to consider more than one concept and could easily be addressed with current querying languages (e.g. SPARQL) [7]. Considering the support for mapping correspondences, we ask which predefined types are supported (Q2). Our data showed that majority of analysed applications (61%) support only the equivalence relation. Other popular mapping types are subsumption and incompatible. The majority of applications (> 64%) don't support the extensibility of predefined mapping types (Q3). Only analyzed APIs (e.g. FOAM [25], Alignment API [26]) could (at least theoretically) be extended to support other mapping types. However, that isn't a flexible and user-friendly approach. Another aspect is the support of functions to express complex mapping. Our data showed that no mapping instance supports conversion functions (G3), e.g. numerical, string or date. An exception is RIMOM [27], which support basic numerical functions to manipulate attributes, e.g. `<user>#addr+#zip=#ci</user>`. Also most representation instances (> 76%) don't support functions for structural rearrangements. Only MAFRA [18], OMT, RIMOM [27] support adding of instances and attributes. Overall complex mappings can not be represented with the analyzed tools. Please note that Alignment API provides an export in XSLT, which supports complex transformations but in the current version none are supported.

### B. Results of G2: Computationally Efficient to Process

On examination of the compatibility of the ontology mapping representation (Q1) we note that 68 % of the representations are implementation independent because they are based on common standard technologies like XML or RDF. Only 27 % of the applications use a proprietary format (text files). The majority of representations (> 36.5 %) are based on RDF. In particular 5 of these 8 mapping representation instances are based on the RDF based INRIA format [26]. This shows that INRIA is still not a de-facto standard but a most popular method for representing mappings. However, this popularity is supported by the fact that the Ontology Alignment Contest demands that all results are delivered in the INRIA format [24]. Considering the tool support (Q2), to the best of our knowledge the majority of mapping representations can only be edited and visualized in their original tools. The only exception is the INRIA format which can be processed by different mapping tools, e.g. Lily, FOAM, RIMOM. Also, as far as we know none of the analyzed applications provide any sophisticated management or sharing tools for mapping information. One exception is COMA++ which provides functions for the manipulation of previous confirmed mapping results, e.g. invert domain or difference analysis [19]. Also worth mentioning is the OMT which supports the automatic testing of mappings to ensure that a given set of source instances translate into the expected set of target instances. However, many tools are based on standard languages like XML or RDF and these can be processed by other common applications.

### C. Results of G3: Enable Sharing and Reuse of Existing Mappings

The third goal analyzed was: how mapping representations instances support the sharing and reuse of previous mappings [7,8]. In particular we analyzed how the ontology mapping life cycle is documented. The first question was: what meta-data is supported to document the source and target ontologies (Q1). The first finding is that all mapping representations contain an ontology identifier, e.g. URI, file paths or labels. This is not surprising because a basic requirement for any mappings is the ability to identify the source and target ontology. Furthermore, none of the analyzed representations provide any information on the version of the processed ontologies. This is quite odd because ontologies can be very dynamic (reviews, updates) and the validity of mappings must be checked for any new version of the ontology. For the processing and especially the applicability of automated matching algorithms it is important to know the ontology format. However, only 64 % of the representations provide such information and in the majority the format can only be deduced by the file extension. This is ambiguous because ".owl" could indicate an OWL DL or OWL Full ontology, e.g. which is essential for a reasoning based matcher. Only the INRIA format [12] and the XML format used by OMT explicitly specify the format which is more appropriate for users and applications. None of the analyzed mapping representations contained information on the used canonical format, the terms usage as well as ontology measurements.

The second question is: how is the mapping phase documented (Q2)? It is important to understand how matching candidates were created to decide if a mapping can be reused. Most applications don't provide any information on the applied matching policies. Only in FOAM [25], an individual classifier can be defined to model simple matching policies. Considering information on the applied type of matcher we must note that only FOAM explicitly specifies if an automated or manual matching was applied. This is quite odd given the fundamental difference between automatic and manual matching relating to quality and quantity [11,17]. In addition no representations provided details on the manual matching process (e.g. who, when) which makes a validation almost impossible. On the other side at least 23 % of the mapping formats specified the applied matching algorithm but commonly by unambiguous labels, e.g. "Value Algorithm" in Ontobuilder. In common quite different parameters are used to configure automated matchers [11] and it is surprising that only 9 % of the representations specify the applied matcher configuration. Also none of the formats contain information on the specific type of automated matching algorithm, e.g. string or structure based.

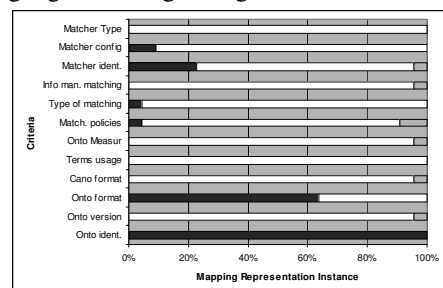


Fig. 1. Results for the Documentation of the Source and the Target Ontologies Phase and the Mapping Phase

The next question was: how is the mapping phase documented in the mapping representation? (Q3). Our results showed that most mapping representation instances don't provide information on applied mapping policies. Only in MAFRA is it possible to define simple mapping conditions. In addition, most formats don't provide information on pre-validated user mappings. Only the representation used by FOAM provides an explicit link to pre-validated user mappings. Also no information is available on the context in which the mapping was created. This is especially problematic because many decisions made in the mapping life cycle are based on external factors and therefore the context is essential for a validation of mappings. The majority of representations (59 %) provide a confidence level for each mapping pair. In common this is a normalized measure of the strength of the relation provided by the applied matching methods [11]. However, it is problematic that no information is available about, how these individual ratios are calculated and should be interpreted. Also the documentation of the applied mapping strategy is very limited. Only FOAM and RIMOM provide a placeholder for a mapping strategy.

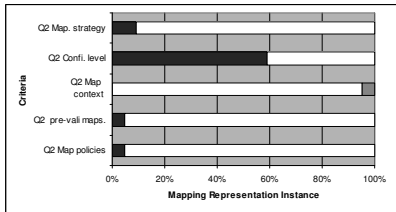


Fig. 2. Results for Q2 Documentation of Mapping Phase

In conclusion we asked: how is the management phase documented in the mapping representation? (Q4). None of the analyzed mapping representations provided any information on the distribution system and therefore it is difficult to find the newest version of the mappings. Also no information on the version of the mapping itself is provided as well as no information on possible conflicts. Really surprising was that no representation contained information on the author or the date of creation. The only exception is RIMOM which at least stores a creation date but it is unclear how the value should be interpreted. To know which specific mapping format used is essential for processing, but only 50 % of the mapping representations explicitly specify their format. In addition none of the formats provide any information on the authority for changes, relevant dependencies or the method for change propagation. Overall this is a major problem because it makes the management and sharing of mappings over time or in a different context challenging especially if the source and target ontologies evolve over time. As a result, current mapping phase relies on external change management and consistency systems.

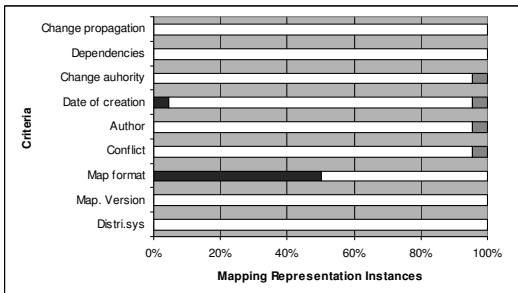


Fig. 3. Results for Q3 Documentation of Management Phase

The last question was: how is the interpretation of the meta-data supported? (Q4). Our data showed that the majority of representations (> 73%) use URIs to identify elements in the mapping representation. However, URIs are commonly established but they are not consistently used in all implementations. Thus not always an unambiguous identification is provided, e.g. only 68 % of the source and target ontologies are identified by a URI, the rest only by file paths and simple labels. In addition, in all mapping representations human-readable labels can be found which helps users to interpret the mapping representations. However, only 59 % of the applications provide a documentation which is commonly very rudimentary. Also none of the URI's refer to a explaining web resource.

#### IV. SUMMARY AND FUTURE WORK

In this paper we have presented an evaluation of ontology mapping representations. In summary, our evaluation revealed three insights. Firstly, when the heterogeneity of mapping use-cases [28] is considered, the level of supported expressiveness in mapping applications and representations is still too low and can not be extended flexibly. The majority of representations support only equivalence relation and no complex mappings. This result is not surprising because most applications are designed as matching tools and their main purpose is the identification of equivalence relations. However, this study is limited to actual implementations but other more generic and expressive mapping languages have been designed, e.g. C-OWL [29]. Such languages may be more powerful but they are not currently used or supported by tools.

Secondly, our evaluation showed that all phases of the ontology mapping lifecycle are very poorly documented and as a result management and reuse of mappings is insufficiently supported by current mapping representations. The lack of meta-data makes it impossible to identify the provenance of the mappings, the latest version of the mappings or the context in which they were created or used. Also disappointing is the common lack of sufficient documentation which makes correct and consistent interpretation of the representations difficult or impossible. Another disadvantage is that currently all meta-data is stored in single attributes only. However, most meta-data elements have a complex knowledge structure and a simplified model is not enough for a practical reuse, e.g. an authors name is not enough to contact him. Recently, sophisticated technologies have been evolved to model complex meta-data, e.g. FOAF for contact details. Such common and established technologies are currently not used in mapping representations but could support the creation of richer knowledge models and interoperability of lifecycle meta-data.

Thirdly, the evaluation showed that the majority of mapping representations can be processed efficiently because they build on standard technologies. RDF is the most common language to express mappings and the RDF based INRIA format has the highest chance of establishment as a de facto standard. However, the majority of the representations can only be reused efficiently in the original application which makes reuse in a broader scale challenging. Hence, ontology mapping representations are still very limited and heterogeneous in terms of expressiveness and meta-data support. No standardised ontology mapping representation has yet emerged or is generally accepted.



The reasons for this are the multitude of mapping/matching approaches available, e.g. different matching algorithms, matching types etc. Each approach has unique requirements for mapping representation, simply because different information and structures need to be represented to express a correspondence. The design of a mapping representation which fulfils all those requirements might be too complex or could lead to a format which represents only the smallest common denominator. For example INRIA is generic but, compared to proprietary formats (e.g. FOAM) less detailed. Multiple mapping representations may be unavoidable because for different mapping scenarios, different representations of the mapping correlations are suitable. In contrast, meta-data which documents the mapping lifecycle is more uniform and for most correlation representations are available. As a result we propose that it is more beneficial to develop the concept of a flexible enrichment of existing and future ontology mapping representations in order to augment their usage, reuse and management. In particular, in an ontology based meta-layer a common vocabulary for modelling life-cycle meta-data could be established and linked to the individual formats representing mapping correlations [20]. Established mapping formats and tools don't need to be changed but available meta-data can still be stored and retrieved in a structured, documented and predictable way.

In conclusion, the remarkable efforts to support the creation of ontology mappings are just the first step. Further research is needed to develop more powerful concepts for the management, sharing and reuse of ontology mappings to even begin to support the flexible communication of a common understanding of a domain at a scale large enough to control the overall information glut [1].

#### ACKNOWLEDGEMENTS

This work is partially funded through the Science Foundation Ireland FAME project (award No. 08/SRC/I1408).

#### REFERENCES

- [1] Antoniou, G., van Harmelen, F., A Semantic Web Primer (Cooperative Information Systems), The MIT Press, 2004.
- [2] Gruber, T. A. Transistional Approach to Portable Ontology Specifications, Knowledge Acquisition, 5, pp. 199-220, 1993.
- [3] Fensel, D., Ontologies Silver Bullet for Knowledge Management and Electronic Commerce, 2nd edition, Berlin, 2003.
- [4] Corcho, O. A declarative approach to ontology translation with knowledge preservation, Volume 116 Frontiers in A.I., 2005.
- [5] Euzenat, J., An API for ontology alignment, in: Proceedings of the International Semantic Web Conference (ISWC 2004), Springer, Berlin, Germany, 2004. pp. 698-712.
- [6] Pepijn, R. S. V., Dean, M. J., Bench-capon, T. J. M. , Shave, M., An analysis of ontological mismatches: Het-erogeneity versus interoperability, AAAI, Spring Symposium on Ontological Engineering, Stanford, USA, 1997.
- [7] Bouquet, P., Ehrig, M., Euzenat, J. et al., D2.2.1 Specification of a common framework for characterizing alignment, <http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/deli/kweb-221.pdf>, 2005.
- [8] O'Sullivan, D., Wade, V., Lewis, D. Understanding as We Roam, in IEEE Internet Computing, 11, (2), 2007, p26 - 33 DOI: <http://doi.ieeecomputersociety.org/10.1109/MIC.2007.50>
- [9] Hameed, A, Preece, A., Sleeman, D., Ontology Reconciliation, Handbook of ontologies, in: Stabb S. and Suder R. (eds) International Handbooks on Information Systems, Springer Verlag, Berlin, Germany, 2004, pp 31-250.
- [10] Kalfoglou, Y., Schorlemmer, M. Ontology mapping: the state of the art. in The Knowledge Engineering Review, 18(1):1-31, 2003.
- [11] Euzenat et al. D2.2.3: State of the art on ontology alignment, <ftp://ftp.inrialpes.fr/pub/exmo/reports/kweb-223.pdf>, 2004. E04b

- [12] Euzenat et al. D2.2.6: Specification of the delivery alignment format, 2006. <http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/deli/kweb-226.pdf>
- [13] Shvaiko, P., Euzenat J., A Survey of Schema-based Matching Approaches, in DIT Technical Report DIT-04-87, 2004.
- [14] Noy, N., Semantic Integration A Survey of Ontology-Based Approaches, in Special Issue on Semantic Integration, SIGMOD Record, Volume 33, Issue 4, pages 65-70, December 2004.No04
- [15] Basili, V. R., Caldiera, G., Rombach, H. D., Goal Question Metric Approach, <ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf>, 2000.
- [16] Thomas, H., O'Sullivan, D., Brennan, R.: Evaluation of Ontology Mapping Representations. In: Workshop on Matching and Meaning, Edinburgh, 2009
- [17] Falconer, S., Storey, M.-A., A cognitive support framework for ontology mapping. In Proc. of the 6th Int. Semantic Web Conference, <http://iswc2007.semanticweb.org/papers/113.pdf>, 2007.
- [18] Maedche, A., Motik, B., Silva, N. et al. MAFRA - A MAPPING FRAMework for Distributed Ontologies, in: Proc. of the 13th Int. Conf. on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 2002, pp. 235 - 250. Ma02c
- [19] Aumüller, D. et al. Schema and Ontology Matching with COMA++. In: Proc. of the ACM SIGMOD Int. Conference on Management of Data, 2005; pp. 906-908.
- [20] Thomas, H., Redmann, T., Markscheffel, B. Controlled semantic tagging, in: Shoniregun, C. A., Logvynovskiy, A.: Proc. of the International Conference on Information Society, 2007, pp. 346-352.
- [21] Beigi, M., Calo, S., Verma, D., Policy Transformation Techniques in Policy-based Systems Management, in: Proc. of IEEE Policy 2004, NY, June 2004.
- [22] Smith, B., Ontology and Information Systems, Lecture Text, [http://ontology.buffalo.edu/ontology\(PIC\).pdf](http://ontology.buffalo.edu/ontology(PIC).pdf), 2000.
- [23] Garshol, L.M., Moore, G. ISO/IEC JTC1/SC34, Information Technology, <http://www.isotopicmaps.org/sam/sam-model/> (2006)
- [24] Wang, P. Xu, B. Lily. Proc. of the 3<sup>rd</sup> int. workshop on Ontology Matching,08,[http://dit.unitn.it/~p2p/OM2008/oaie08\\_paper7.pdf](http://dit.unitn.it/~p2p/OM2008/oaie08_paper7.pdf)
- [25] Ehrig, M, Sure, Y. FOAM - Framework for Ontology Alignment and Mapping; In: Ashpole, B, et al: Proceedings of the Workshop on Integrating Ontologies, volume 156, pp. 72-76. October 2005.
- [26] Euzenat J., An API for ontology alignment, Int. Semantic Web Conference 2004, Springer, Berlin, Germany, 2004, pp 698-712.
- [27] Caracciolo, C., et al.Results of the Ontology Alignment Evaluation Initiative 2008. Proc. of the 3<sup>rd</sup> int. workshop on Ontology Matching, 2008. [http://www.dit.unitn.it/~p2p/OM-008/oaie08\\_paper0.pdf](http://www.dit.unitn.it/~p2p/OM-008/oaie08_paper0.pdf).
- [28] Giunchiglia, F., Shvaiko, P., Semantic matching, in: proc. of the Workshop on ontologies & distributed systems, pp. 193-146, 2003.
- [29] Bouquet P. et al. C-OWL: Contextualization Ontologies. In proc. of the 2<sup>nd</sup> Semantic Web Conf. 2003, pp. 164-179

#### APPENDIX A OVERVIEW OF EVALUATED APPLICATIONS

Application	Link
Alignment API	<a href="http://alignapi.gforge.inria.fr/">http://alignapi.gforge.inria.fr/</a>
Anchor-PROMPT	<a href="http://protege.stanford.edu/plugins/prompt/prompt.html">http://protege.stanford.edu/plugins/prompt/prompt.html</a>
COMA++	<a href="http://dbs.uni-leipzig.de/Research/coma">http://dbs.uni-leipzig.de/Research/coma</a>
Context Matching Algorithm (CtxMatch)	<a href="http://dit.unitn.it/~zanobini/downloads.html">http://dit.unitn.it/~zanobini/downloads.html</a>
CROSI Mapping System	<a href="http://www.aktors.org/crosi/">http://www.aktors.org/crosi/</a>
Falcon-AO	<a href="http://iws.seu.edu.cn/projects/matching/projects.jsp">http://iws.seu.edu.cn/projects/matching/projects.jsp</a>
Framework for Ontology Alignment & Mapping (FOAM)	<a href="http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/">http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/</a>
Lily	<a href="http://ontomappinglab.googlepages.com/lily.htm">http://ontomappinglab.googlepages.com/lily.htm</a>
MAFRA	<a href="http://mafra-toolkit.sourceforge.net">http://mafra-toolkit.sourceforge.net</a>
MapOnto	<a href="http://www.cs.toronto.edu/semanticweb/maponto/">http://www.cs.toronto.edu/semanticweb/maponto/</a>
OntoBuilder	<a href="http://iew3.technion.ac.il/OntoBuilder">http://iew3.technion.ac.il/OntoBuilder</a>
Ontology Mapping Tools	<a href="http://www.wsmx.org/">http://www.wsmx.org/</a>
Risk Minimization Ontology Mapping (RiMOM)	<a href="http://keg.cs.tsinghua.edu.cn/project/RiMOM/">http://keg.cs.tsinghua.edu.cn/project/RiMOM/</a>

# Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas

Stefan Biffl, Wikan Danar Sunindyo, Thomas Moser

*Institute of Software Technology and Interactive Systems, Vienna University of Technology  
Favoritenstrasse 9-11/188, Vienna, Austria  
{ stefan.biffl, wikan.sunindyo, thomas.moser }@tuwien.ac.at*

**Abstract**—Stakeholders from several domains with local terminologies have to work together to develop and operate software-intensive systems, like production automation systems. Ontologies support the translation between local terminologies via common domain concepts. Unfortunately, the ontology models can become large and complex if they include several aspects on a domain and some parts of the data model are volatile. In this paper, we propose a data modeling approach to support ontology users based on ontology building blocks, so-called “Ontology Areas” (OAs), which allow solving tasks with smaller parts of the overall ontology. We evaluate the proposed approach with use cases from the production automation domain: translation between stakeholder roles to support design-time and run-time decision making. Major result in the study context is that OAs improved the efficiency of data collection for decision making.

## I. INTRODUCTION

The integration of business processes and IT systems in homogeneous environments (i.e., consistent data formats and terminology) is supported by well-established approaches like data integration using Scheer’s ARIS for CIM [21]. However, in more heterogeneous environments with a range of data formats and local terminologies like the production automation domain, typically stakeholders from several areas (e.g., business experts, software engineers and electrical engineers) work together to develop and operate software-intensive systems. A homogenization of these environments is often not achievable, if the stakeholders come from different organizational backgrounds or organizations change over time due to mergers and acquisitions. The precondition for successful semantic integration is a common understanding on the relevant concepts in the problem domain of the project.

An example for a collection of common problem domain concepts is the *Enterprise-Control System Integration*<sup>1</sup> (ECSI) standard [1] for developing automated interfaces between enterprise and control systems. The objectives of ECSI are to provide a) a consistent terminology as foundation for supplier and manufacturer communications, b) consistent information models, and c) consistent operations (process) models, which are the basis for clarifying application functionality and how information shall be used.

However, a standard like ECSI can only cover parts of the problem domain without getting too complex and hard to use. Further, many key players in the production automation domain currently do not follow this standard, which often hinders the cooperation of stakeholders in projects, since trans-

formations between stakeholder terminologies to overcome semantic gaps between the stakeholders need to be conducted by scarce experts or carefully hand-crafted.

Ontologies are flexible open-world data models for knowledge representation, which store information in machine-understandable notation [12]. Therefore, ontologies can help to bridge semantic gaps between partial data models by providing mappings between them via common domain concepts. Ontologies usually capture problem-domain-specific information which can be reused later. Due to their concurrent development ontologies need to be checked for inconsistencies to stay useful. However, ontologies in practice usually have to combine several view points and thus get large and complex, particularly, if the ontology contains volatile domain elements, such as run-time data.

In this paper, we propose a data modelling approach that helps structure ontologies with ontology building blocks, so-called “*Ontology Areas*” (OAs). An OA is a meaningful part of an ontology for a stakeholder, which helps ontology users managing a complex ontology. The combination of all needed OAs represents the overall ontology for supporting the original engineering process.

We evaluate the proposed OA approach with use cases in the production automation domain: 1. Translation between local stakeholder terminologies; 2. Provision of design context for run-time data interpretation; and 3. Run-time measurement representation for design model improvements. The use cases are based on the data model of the “*Simulator for Assembly Workshops*” (SAW) [15] and compare the performance of an ontology with and without OAs. The evaluation showed that OAs made the data collection in the ontology for decision support more efficient in the study context, since the OAs result in a smaller ontology for the tasks in the use cases.

The remainder of this paper is structured as follows: Section 2 summarizes related work on system integration and ontologies. Section 3 describes the industry use case and Section 4 derives research issues. Section 5 introduces the OA approach, while Section 6 evaluates the approach and discusses the results. Finally, Section 7 concludes the paper and identifies further work.

## II. RELATED WORK

This section summarizes related work on system integration and ontologies for semantic integration to reconcile different views of stakeholders on system data.

<sup>1</sup> <http://www.isa-95.com>

### A. Integration of Heterogeneous Systems

System integration is the task to combine a range of smaller systems to appear as one big system. There are several levels at which system integration could be performed [3], but there is so far no standardized integration process that explains how to integrate systems in general.

Typical integration solutions focus either on technical heterogeneity (how to connect systems that use different platforms or protocols) or on semantic heterogeneity (how to translate data in messages between systems that use different data formats or terminologies). In order to cope with technical heterogeneity on service level middleware technology [9] supports syntactical transformation between services, while the semantic heterogeneity of services can be addressed with a common data schema [13]. Limitations of these integration approaches are: 1. The need for a common data schema [13], which is hard and time-consuming to negotiate, sometimes impossible if stakeholders continue to disagree. 2. The need for integration over heterogeneous middleware technologies (with different APIs or network architecture styles) implies the development of static and therefore inflexible wrappers between each combination of middleware technologies, and thus increases the complexity of communication.

Semantic integration is defined as the solving of problems originating from the intent to share data across disparate and semantically heterogeneous data [13]. These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modelling of complex relations in different sources [20]. Over the last years, semantic integration became increasingly crucial to a variety of information-processing applications and has received much attention in the web, database, data-mining and AI communities [6]. One of the most important and most actively studied problems in semantic integration is establishing semantic correspondences (also called mappings) between vocabularies of different data sources [7].

### B. Ontologies for Semantic Integration

An ontology is a representation vocabulary for a specific domain or subject matter, like production automation. More precisely, it is not the vocabulary as such that qualifies as an ontology, but the (domain-specific) concepts that the terms in the vocabulary are intended to capture [5]. Many authors like Goh [11] identified three main categories of semantic heterogeneities in the context of data integration that can appear: confounding conflicts (e.g., equating concepts are actually different), scaling conflicts (e.g., using different units for the same concept), and naming conflicts (e.g., synonyms).

Noy [19] identified three major dimensions of the application of ontologies for supporting semantic integration: the task of finding mappings (semi-)automatically, the declarative formal representation of these mappings, and reasoning using these mappings. There exist two major architectures for mapping discovery between ontologies: 1. It is possible to create a general upper ontology which is agreed upon by developers of different applications. Two examples for ontologies that are built specifically with the purpose of being formal top-level

ontologies are the *Suggested Upper Merged Ontology* (SUMO) [18] and DOLCE [10]. 2. There are approaches comprising heuristics-based or machine learning techniques that use various characteristics of ontologies (e.g., structure, concepts, instances) to find mappings. These approaches are similar to approaches for mapping XML schemas or other structured data [4, 6]. The declarative formal representation of mappings is facilitated by the higher expressive power of ontology languages which provide the opportunity to represent mappings themselves in more expressive terms.

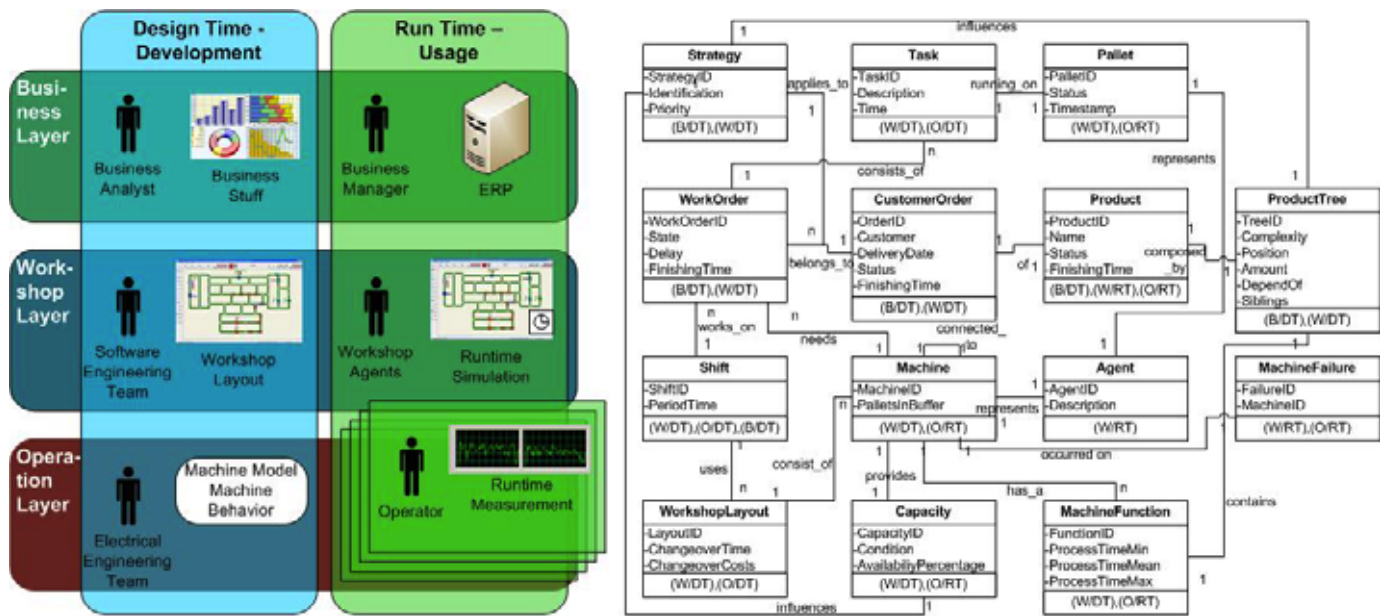
Uschold and Gruninger [23] identified four main categories of ontology application to provide a shared and common understanding of a domain that can be communicated between people and application systems [8]: Given the vast number of non-interoperable tools and formats, a given company or organization can benefit greatly by developing their own neutral ontology for authoring, and then developing translators from this ontology to the terminology required by the various target systems. While it is safe to assume there will not be global ontologies and formats agreed by all possible stakeholders, it is nevertheless possible to create an ontology to be used as a neutral interchange format for translating among various formats. There is a growing interest in the idea of “Ontology-Driven Software Engineering” in which an ontology of a given domain is created and used as a basis for specification and development of some software [19]. The benefits of ontology-based specification are best seen if there is a formal link between the ontology and the software. To facilitate search, an ontology is used as a structuring device for an information repository (e.g., documents, web pages, names of experts); this supports the organization and classification of repositories of information at a higher level of abstraction than is commonly used today.

As alternative approach for semantic integration of system models the infrastructure of Model-Driven Architecture (MDA) [16] provides architecture for creating models and meta-models, defining transformations between these models, and managing meta-data. Although the semantics of a model is structurally defined by its meta-model, the mechanisms to describe the semantics of the domain are rather limited compared to machine-understandable representations using, e.g., knowledge representation languages like RDF<sup>2</sup> or OWL<sup>3</sup>. In addition, MDA-based languages do not have a knowledge-based foundation to enable reasoning (e.g., for supporting quality assurance), which ontologies provide [2]. Beyond traditional data models, like UML class diagrams or entity relationship diagrams, ontologies provide methods for integrating fragmented data models into a common model without losing the notation and style of the individual models [14].

Seidenberg and Rector [22] proposed web ontology segmentation to counter decreasing ontology performance when ontology size increases. The algorithm to make ontology segmentation is similar to our approach, but we extend the usage of ontology areas for more stakeholders and volatilities.

<sup>2</sup> Resource Description Framework: <http://www.w3.org/RDF/>

<sup>3</sup> Web Ontology Language: <http://www.w3.org/2007/OWL>



**Figure 1: Sources of semantic gaps between stakeholders: domain layers, design-/run-time views; the data model contains common domain concepts to bridge semantic gaps.**

### III. INDUSTRY USE CASE

In cooperation with industry partners in the production automation domain we conducted the project “Simulator for Assembly Workshops” (SAW) [15], which simulates complex reconfigurable production automation systems by scheduling sequences of transport and machine tasks over 100 times faster than the actual hardware<sup>4</sup>. The SAW simulator has been validated with real hardware components to ensure simulation validity for real-world production automation systems. In the SAW context stakeholders from different backgrounds work together and could benefit from better automated access to each others data models which is currently only possible via the stakeholders themselves as the data models are not well integrated.

Figure 1 illustrates sources of semantic gaps between stakeholders: stakeholder domain layers with different local terminologies; and design-/run-time views which are semantically not well connected. The data model, in our case an ontology model (the Engineering Knowledge Base (EKB) [17]), contains common domain concepts to bridge the semantic gaps between stakeholder terminologies and design-/run-time views.

The three stakeholder layers in Figure 1 are: a) the *business layer (B)* for production planning to fulfil customer orders by assigning optimal work orders to the workshop; b) the *workshop layer (W)* for coordinating the complex system of transport elements and machines to assemble smaller basic products into larger more comprehensive products according to the work orders; and c) the *operation layer (O)* for monitoring the individual transport system elements and machines to ensure their contributions to the workshop tasks. Those three layers

are divided into two parts based on the time those layers worked on, namely design time (development) and run time (usage).

Figure 1 (right hand side) illustrates part of the data model that represents common domain concepts for the uses cases in UML-class-diagram style notation. The bottom box of each data element shows which stakeholder layer (B, W, and O) needs this data element to conduct their tasks and when: at Design Time (DT) or Run Time (RT).

From the SAW project we derived the following use cases that illustrate semantic gaps between stakeholders and how to overcome these gaps using ontology-based approaches.

#### UC-1. Translation between local stakeholder terminologies.

The business manager on the business layer receives customer orders and schedules work tasks to the coordinator in the workshop layer. While they have a defined interface for exchanging work task information, they use local terminologies for concepts that are only occasionally needed to resolve scheduling issues, e.g., reference to specific customer orders if limited workshop capacity does not allow to fulfil all work tasks in a shift and negotiation on which tasks have higher priority are necessary to determine which customer orders will be fulfilled. Because the stakeholders use different terminologies, translations are necessary to automate references to customer orders between stakeholders in business and workshop layers.

#### UC-2. Run-time measurement data representation and analysis for design model improvements.

If an engineering knowledge base is available to support run-time decisions with design knowledge, it is easy to also provide all kinds of run-time measurements linked to design elements, e.g., actual capacity of infrastructure, to iteratively improve the accuracy of design estimates with feedback from run time.

<sup>4</sup> Automation & Control Institute; <http://www.acin.tuwien.ac.at>

#### IV. RESEARCH ISSUES

The general idea of Ontology Areas (OAs) is to structure a comprehensive ontology into smaller building blocks with the following benefits for the designer and user of the ontology:

- A *smaller ontology* based on OAs that contains the minimal necessary knowledge for a specific task can be selected from a comprehensive ontology to facilitate more efficient use and change.
- We expect a smaller ontology (consisting of selected OAs) to exhibit *lower cognitive complexity* for designers who work with ontologies to make tools that support the automation of stakeholder tasks.
- Specific OAs can contain the more volatile ontology elements and thus make the design of the overall ontology *more stable against changes*.

As measurement criteria for evaluation we use the size of an ontology (and an OA) by counting the number of facts and relationships. In our study context the comprehensive ontology consists of: a) the production automation domain concepts (i.e., data model in Fig. 1) for design-time and run-time elements; and b) stakeholder extensions to the data model, such as local terminologies and mappings, for all stakeholders.

We used the following guidelines to design the OAs: a) concepts that a particular stakeholder (in business, workshop, or operation layer) needs to fulfil his typical tasks in order to achieve cohesiveness of the OAs; b) discern between common domain concepts and local add-ons of a stakeholder (such as terminology), which may change in different project contexts; c) keeping apart more stable design-time concepts from more volatile run-time concepts; and d) structuring volatile run-time data by manageable time intervals depending on the frequency of data elements' change. According to these guidelines examples for concrete OAs are: the design-time concepts of a business stakeholder and the run-time terminology of a workshop stakeholder.

From the use cases we derive the following research issues (RIs) to investigate the benefits of an ontology structured with OAs compared to an ontology without OAs.

**UC-1. Translation between local stakeholder terminologies.** The ontology supports each role by allowing to use their local terminology to communicate with other stakeholders. For this task sufficient OAs need to contain for the communicating stakeholders: the common domain concepts in their universe of discourse (see also in Fig. 1 the data elements and their link to associated stakeholders), local terminologies, mappings between local terminology elements and common domain concepts (on class level).

**RI-1a:** Compare the *complexity* (size) of the minimal ontology with OAs to the complexity of the overall ontology in the study context.

**RI-1b:** Compare the *efficiency* of the minimal ontology with OAs to the efficiency of the overall ontology in the study context to conduct the translation task.

The other use cases address benefits from making links between design-time and run-time data elements available at run time.

#### **UC-2. Run-time measurement data representation and analysis for design model improvements.**

In the study context the collection of run-time data points, e.g., on process characteristics and quality of service of the infrastructure, helps to provide data for future design improvements, e.g., for more realistic planning and more efficient system configurations. The designers and quality management personnel, who conduct the data analysis procedures, often do not know in advance precisely which analysis functions they will need. Thus, a considerable amount of raw data would be beneficial to store in the ontology for querying design-time relationships and run-time data together. Unfortunately, even moderate data collection (10 data points) at reasonable frequency (e.g., one measurement every second) leads over a shift of 8 hours to a number of run-time data elements that easily exceeds the size of the design-time data elements in the ontology.

OAs that are designed to hold all measurement instances of a data element in a certain time interval (e.g. one minute) allow to keep the complexity of the ontology needed for analysis manageable: Only the OAs that contain relevant run-time measurements for a given analysis need to be considered.

**RI-2a:** Determine the *minimal complexity of OAs* to support a specific data analysis task more efficiently, such as calculating process characteristics. Compare the result with OAs to the (cognitive) complexity of using a whole ontology.

**RI-2b:** Compare the *efficiency* of the minimal ontology with OAs to the efficiency of the overall ontology in the study context to conduct the data analysis task

#### V. ONTOLOGY AREAS FOR BRIDGING SEMANTIC GAPS

In this Section we explain in more detail how to address the use cases with an ontology that uses OAs as basis for the evaluation of the RIs in Section 6.

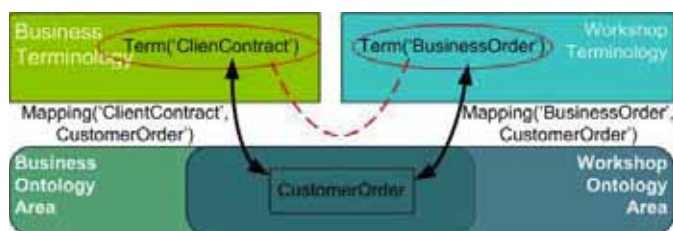
An ontology area is a subset of ontology as a building block that can solve a certain task. The ontology can be broken into ontology areas based on several aspects, for example by the time, volatility, layer and roles. Figure 1 shows the break down of ontology into several ontology areas based on the stakeholder layers (business, workshop, operation) and time when models are mostly used (design time and run time). Some parts of the data mode are much more volatile than others, e.g., run-time process measurements compared to design-time workshop layout. For example, each data point measured once a second in a shift that takes 8 hours produces around 30,000 data point instances, which need to be reduced by statistical methods or will take considerably storage space.

To make an OA from the whole ontology, we can follow this basic algorithm. First, define a task that is needed to be solved by the stakeholder. Second, find related classes for doing the task. Third, find classes that linked to the classes in step two. Fourth, drop other classes that are not needed and save as a new ontology. Also, we can reconstruct the whole ontology from the ontology areas, by merging them together into one ontology by using ontology tool like Protégé.

We illustrate in three use cases (UC-1 to UC-2) how OAs help reduce the complexity of the ontology for bridging semantic gaps in production automation systems.

**UC-1. Translation between local stakeholder terminologies.** The stakeholders of the production automation systems need to work together to achieve their goal. A common data schema is not possible because the stakeholders usually use different data formats, local terminologies and tools to access the data from the system. The ontology (EKB – Engineering Knowledge Base) plays a role as a common domain concept, where the local terminologies from the stakeholders will be mapped to. By mapping each local terminology to the ontology, the system can translate the local terminologies from one stakeholder to the other stakeholders. The translation could be the name of function, some names in the argument of the function, different data format, or the meaning of some parameters. However, the complexity of the ontology may increase when the number of the terminologies and the stakeholders is also increases, since the ontology should store all terminologies, the mappings and the common concepts.

By using the ontology areas, the stakeholder can take a small part of the ontology that he really cares and solving his task with the same results but less complexity than by using the full ontology. The example is illustrated on figure 2.



**Figure 2. Translation between Business Terminology and Workshop Terminology.**

The business stakeholder has a local terminology “ClientContract”, while the workshop stakeholder has a local terminology “BusinessOrder”. Both have a common concept to class CustomerOrder in the Ontology Areas. Then, both terminologies will be mapped to the class CustomerOrder as mention in Listing 1a.

**Listing 1a. Mapping terminologies to the common concept.**

```
mapping('ClientContract', 'CustomerOrder').
mapping('BusinessOrder', 'CustomerOrder').
```

From the mappings above, we can have a translation between two local terminologies by using rule on Listing 1b. The query and result can be seen on Listing 1c.

**Listing 1b. Simple translation rules.**

```
translate(Term1, Term2) :-
mapping(Term1, CommonConcept),
mapping(Term2, CommonConcept),
not(Term1 = Term2).
```

**Listing 1c. Translation result.**

```
translate(X, Y).
X = 'ClientContract'
Y = 'BusinessOrder'
```

The translation is just one example for translations in general. OAs for this use case would just consider the parts of the ontologies for the stakeholders involved (see Figure 2): stakeholder concepts, their local terminologies and mappings, which can more easily be added to and removed from an ontology as stakeholders change in a particular context. The evaluation for this use case will be explained on section 6.

**UC-2. Run-time measurement data representation and analysis for design model improvements.** Run-time measurement information can be used to make design time information more accurate. Volatile information like run-time measurement can produce large amounts of data which would make a single ontology unnecessary large and slow down the performance of the ontology. The need for storing a high volume of run-time measurement data in the ontology occurs if the concrete future statistical analysis procedures are not known at the time of measurement.

Partitioning of the ontology in areas of similar volatility allows building partial ontologies for the task or query at hand. Run-time measurement at the frequency of 1 data point per second provides 30,000 data points of shift of 8 hours. If this is too much information for the ontology to hold, it is possible to define OAs for smaller time windows, which allow including the data for a certain time frame to be loaded into the ontology for data analysis as needed without exceeding the capacity of the ontology.

Semantic gaps between run-time measurement and design-time information occur when we have data elements from the interface of the machine at run time, but there is no machine-understandable documentation for the design of the interface. To solve this problem, we first give meaning to run-time data that are needed to be stored in the ontology and then provide a link from run-time to design-time semantics.

For example, to find out the maximum process time of certain machine functions, we can measure the process duration of that machine function in one shift, so we collect sufficient and still manageable data. The measurement result is an event named “process” that consists of the id, the batch number, status and timestamp of machine function. Listing 2a shows several measurement results that can be obtained by filtering run time data. The real data themselves is a very long list.

**Listing 2a. Run-time event data with semantic annotation.**

```
% process (machine function id, batch number, status,
timestamp)
process('MF1', 'B-100', 'start', 2009-02-03 T 10:01:06.01)
process('MF1', 'B-100', 'stop', 2009-02-03 T 10:01:06.11)
process('MF2', 'A-200', 'start', 2009-02-03 T 10:01:06.12)
process('MF1', 'B-101', 'start', 2009-02-03 T 10:01:06.13)
process('MF1', 'B-101', 'stop', 2009-02-03 T 10:01:06.21)
process('MF2', 'A-200', 'stop', 2009-02-03 T 10:01:06.24)
```

To calculate the maximum process time of certain machine function, first we should calculate each process time by using predicate “process\_time” to find the difference between the timestamp of “stop” status and the related timestamp of “start” status from the same machine function and batch number, and then keep it in the list using “list\_of\_process\_time” predicate. Then with using the predicate “maxprocess” we will find the

maximum value of process time of certain machine function (MFun) from the list of process time.

#### Listing 2b. Example analysis rule on run-time data.

```
max(X,Y,X) :- X >= Y.
max(X,Y,Y) :- X < Y.
maxlist([X],X).
maxlist([X,Y|Tail],Max) :-
maxlist([Y|Tail],MaxTail),max(X,MaxTail,Max).
process_time(MF,SN,T) :-
process(MF,SN,start,X),
process(MF,SN,stop,Y),
T is Y - X.
list_of_process_time(List,MFun) :-
findall(T,(process_time(MF,SN,T),MF = MFun),List).
maxprocess(MFun,T) :-
list_of_process_time(List,MFun),
maxlist(List,T).
```

For query, for example we want to know the maximum process time of 'MF1'. The result can be seen on Listing 2c.

#### Listing 2c. Result of data analysis.

```
maxprocess('MF1',T).
T = 0.1
```

The machine function entity in design time consists of the id and process time attributes. Usually the values of process time attributes come from estimation, but by using run-time measurement on process time, we can compare the previous design-time estimates to actual run-time data analysis for research on design improvements.

The illustrating example above is simple enough to conduct statistical analysis at run time, but for more complex statistical analyses, a solution for storing large amounts of data in an ontology may be necessary, which would inflate ontology size and decrease the ontology reasoning performance. OAs allow to manage stacks of run-time data elements and keep the size of ontology within well-performing capacity ranges.

## VI. EVALUATION AND DISCUSSION

We have implemented the OAs from the SAW ontology using Protégé 3.3.1. The SAW ontology consists of 24 classes and 3,000 instances from the simulation of production automation system. The evaluation will compare the measurement of the whole ontology and the ontology areas for three different use cases explained in section 5, as follows.

**UC-1: Translation between local stakeholder terminologies.** We compare the complexity (size) of the minimal ontology with OAs to the complexity of the overall ontology in the study context. For the minimal ontology with OAs, the business and workshop stakeholders have local terminologies of 300 and 400 words, respectively. Both need 100 words to communicate with each other. There are 200 to 700 data elements representing common knowledge, and 200 words for mapping from both local terminologies to the common concepts. Totally 1,100 to 1,600 entities are needed for the OAs.

Meanwhile, the comprehensive ontology for 6 stakeholders consists of around 1,800 words for local terminologies and around 300 words to communicate with each other. There

are 1,600 words of common knowledge, and 600 to 1,800 words for mapping of all local terminologies to common concepts. In total, the comprehensive ontology consists of 4,200 to 5,400 words. In this case, OAs can reduce the ontology size to 20 to 30 % of the comprehensive ontology.

We can compare the efficiency of the minimal ontology with OAs to the efficiency of the whole ontology in conducting the translation task as follows. To produce 100 words of translation results from 200 words of mapping, the OAs needs 3 operators of query applying to those mapping.

The comprehensive ontology can produce more translations (300 words) with 3 operators of query as well. But the query should be applied to more mapping (600 to 1,800 words). With OAs we can reduce the size of mapping and make the operation faster.

**UC-2: Run-time measurement and analysis for design improvement.** For evaluation we will determine the minimal complexity of OAs to support a specific data analysis task more efficiently, such as calculating process characteristics. Then we will compare the result with OAs to the (cognitive) complexity using a comprehensive complexity.

In the OAs of the specific task, for 1 volatile entity the run-time measurement consists of 30,000 data points per shift. In the overall ontology, there may be many more, e.g., 300,000, data points in one shift. By using the OAs, the user can focus only on entity that he needs, and thus reduce the complexity of data handling considerably.

The efficiency of the minimal ontology with OAs is compared to the efficiency of the overall ontology in the case to conduct the data analysis task as follows. In the OA, to obtain 5 data points analysis, it needed to run 3 operators of query over 30,000 data points at one shift. Hence 18,000 operations on data points are needed to obtain one of the measurements.

In the whole ontology, to obtain 20 data points analysis, it needed to run 3 operators of query over 300,000 data points at one shift. Hence 45,000 operations on data points are needed to obtain one of the measurements. OA is notably more efficient than overall ontology.

**Lesson learned.** From the experiences with these use cases, we can learn the following lessons.

*Building a smaller ontology for a task.* As OAs allow focusing on the content of interest for a stakeholder task, we could show that the resulting ontology is considerable smaller. A smaller ontology is often also more efficient to handle and allows tackling tasks that use a particularly large number of data elements (e.g., run-time measurements in UC-3).

*Focus stakeholders on relevant data elements.* The combination of OAs, design-time, and run-time data elements allowed filtering relevant data elements for stakeholders, which would not be possible without the combination. Thus the OA approach helped lower the cognitive complexity for stakeholders by providing just the relevant subset of the comprehensive ontology.

*Version management for ontology areas.* With the OA concept we can flexibly build task-oriented ontologies based on different criteria (like volatileness, layers, roles). It is even possible to compare different versions of the same OA (e.g.,

production automation system designed with different parameter settings) to compare the run-time reactions to from changing design parameters. However, this ability also raises the need for better version management for OAs to ensure the building of consistent ontologies for specific tasks.

## VII. CONCLUSION AND FURTHER WORK

Ontologies support the translation between stakeholder local terminologies via common domain concepts, in our case production automation concepts. Typically, the ontology models become very large and complex compared to the basic data model (such as used in a data base to automate run-time processes) if they include several aspects on a domain and some parts of the data model are volatile. In this paper, we proposed a data modeling approach based on ontology building blocks, so-called "Ontology Areas" (OAs), which allow solving tasks with smaller parts of the overall ontology. We evaluated the proposed approach with use cases from the production automation domain. Major result in the study context is that OAs improved the efficiency of data collection task for decision making by lowering the cognitive complexity for designers and users of the ontology.

**Further work.** We see further research in the following directions.

*Effort for OA design and use.* While OAs make a comprehensive ontology, which stores and uses engineering knowledge both at design time and run time, more manageable, their application needs the effort of designers for structuring the overall ontology and for building task-specific smaller ontologies. Thus we will conduct empirical studies on the effort needed to design and use ontologies with OAs. Future work could include human-subject experiments to assess complexity and efficiency more rigorously.

*Guidelines for the OA approach.* While we found OAs useful to manage a large and complex ontology, we see the need for guidelines for the application the OA approach when designing a new ontology as well as for structuring already established ontologies with OAs to improve their performance.

*Maintenance effort.* Particularly for ontologies which should be changed by many users concurrently, we see a potential advantage of the concept of OAs, as areas with different rates of change can be easily separated, simplifying the checking of models for consistency etc. In the context of our case study this could be measuring the effort for typical changes, such as a new workshop layout, new machines, or new connections between machines.

## ACKNOWLEDGMENT

We want to thank our colleagues at ACIN and TU Prague, for their feedback and inspiring discussions; and the SAW team at TU Wien for providing the application environment for the research use case.

## REFERENCES

- [1] American National Standard, "Enterprise-Control System Integration," in *Part 1: Models and Terminology*. vol. ANSI/ISA-95.00.01-2000 North Carolina, USA: ISA (the Instrumentation, Systems, and Automation Society), 2000, p. 142.
- [2] K. Baclawski, M. K. Kokar, P. A. Kogut, L. Hart, J. Smith, J. Letkowski, and P. Emery, "Extending the Unified Modeling Language for Ontology Development," *International Journal of Software and Systems Modeling (SoSyM)*, vol. 1, pp. 142-156, 2002.
- [3] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema, "Developing Applications Using Model-Driven Design Environments," *COMPUTER*, pp. 33-40, 2006.
- [4] S. Bergamaschi, S. Castano, and M. Vincini, "Semantic integration of semistructured and structured data sources," *SIGMOD Rec.*, vol. 28, pp. 54-59, 1999.
- [5] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies, and why do we need them?," *Intelligent Systems and Their Applications, IEEE [see also IEEE Intelligent Systems]*, vol. 14, pp. 20-26, 1999.
- [6] I. R. Cruz, X. Huiyong, and H. Feihong, "An ontology-based framework for XML semantic integration," in *International Database Engineering and Applications Symposium (IDEAS '04)*, 2004, pp. 217-226.
- [7] A. Doan, N. F. Noy, and A. Y. Halevy, "Introduction to the special issue on semantic integration," *SIGMOD Rec.*, vol. 33, pp. 11-13, 2004.
- [8] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*: Springer, 2003.
- [9] E. H. Gail, L. David, C. Jeremy, re, N. Fred, C. John, and N. Martin, "Application servers: one size fits all ... not?," in *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Anaheim, CA, USA, 2003.
- [10] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari, "Sweetening WordNet with DOLCE," *AI Magazine*, vol. 24, pp. 13-24, 2003.
- [11] C. H. Goh, "Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems." vol. PhD: MIT, 1996.
- [12] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," in *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, Eds.: Kluwer Academic Publishers, 1993.
- [13] A. Halevy, "Why your data won't mix," *Queue*, vol. 3, pp. 50-58, 2005.
- [14] M. Hepp, P. De Leenheer, A. De Moor, and Y. Sure, *Ontology Management: Semantic Web, Semantic Web Services, and Business Applications*: Springer-Verlag, 2007.
- [15] M. Merdan, T. Moser, D. Wahyudin, and S. Biffl, "Performance evaluation of workflow scheduling strategies considering transportation times and conveyor failures," in *International Conference on Industrial Engineering and Engineering Management (IEEM 2008)*, 2008, pp. 389-394.
- [16] J. Miller and J. Mukerji, "Model Driven Architecture (MDA)," *Object Management Group, Draft Specification ormsc/2001-07-01, July*, vol. 9, 2001.
- [17] T. Moser, A. Schatten, W. D. Sunindyo, and S. Biffl, "A Run-Time Engineering Knowledge Base for Reconfigurable Systems," Institute for Software Technology and Interactive Systems, Vienna University of Technology, Austria, Vienna, 2009, <http://tinyurl.com/c5snc3>.
- [18] I. Niles and A. Pease, "Towards a standard upper ontology," in *2nd International Conference on Formal Ontology in Information Systems*, 2001, pp. 2-9.
- [19] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *SIGMOD Rec.*, vol. 33, pp. 65-70, 2004.
- [20] N. F. Noy, A. H. Doan, and A. Y. Halevy, "Semantic Integration," *AI Magazine*, vol. 26, pp. 7-10, 2005.
- [21] A. W. Scheer, *Computer-Integrated Manufacturing*, 4th ed.: Springer, 1989.
- [22] J. Seidenberg and A. Rector, "Web Ontology Segmentation: Analysis, Classification and Use," in *International World Wide Web Conference (WWW 2006)* Edinburgh, Scotland: ACM, 2006, p. 10.
- [23] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," *SIGMOD Rec.*, vol. 33, pp. 58-64, 2004.



# LD2SD: Linked Data Driven Software Development

Aftab Iqbal, Oana Ureche, Michael Hausenblas, Giovanni Tummarello  
Digital Enterprise Research Institute (DERI),  
National University of Ireland, Galway  
IDA Business Park, Galway, Ireland  
firstname.lastname@deri.org

## ABSTRACT

In this paper we introduce *Linked Data Driven Software Development* (LD2SD), a light-weight Semantic Web methodology to turn software artefacts such as data from version control systems, bug tracking tools and source code into linked data. Once available as linked data, the related information from different sources is made explicit, allowing for a uniform query and integration. We show the application of LD2SD using a real-world software project as the reference dataset and discuss the added value of LD2SD compared to existing technologies.

## 1. MOTIVATION

In the software development process, both humans and so called *software artefacts* are involved (Fig. 1). Human beings such as developers and clients (customers, project managers, etc.) typically interact not only face-to-face or telephone, but also by means of discussion forums, emails, etc.. The software artefacts shown in the lower half of Fig. 1 can be understood as *heterogeneous, interconnected datasets*, conveying information about the software project and the humans involved.

It is worth mentioning that very often these interconnections are not explicit, hence machine-accessible but rather of an implicit nature (a mentioning of a certain Java class in a blog post, for example). Further, some of these datasets, such as the program source code or versioning data are mainly under the control of a *developer*, whilst other datasets are widely “filled” by *clients*. Then, there are datasets that are shared between developer and clients (e.g., a discussion board). In any case, the datasets are closely related and interdependent. A bug report, for example, may lead to a change in the program code and additionally the documentation needs to be updated. This may be reflected in the configuration management system. Further, a feature request may indirectly arise from a discussion on a discussion board, for example.

Nowadays, development takes place mainly in two environments, (i) the developers Integrated Development Environment (IDE), such as Eclipse<sup>1</sup>, and (ii) the Web, such as for finding examples and documentation, discussions, etc. as a large. We need hence not only make the links between the software artefacts within a project explicit but also al-

<sup>1</sup><http://www.eclipse.org/>

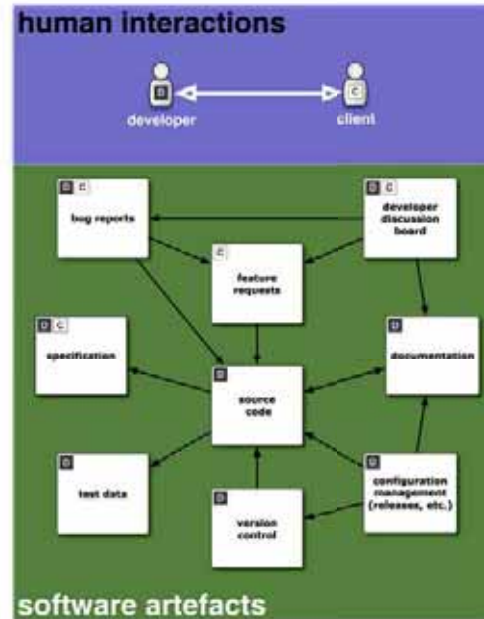


Figure 1: LD2SD Overview.

low connecting to data on the Web. Having such an explicit representation of the connection between the datasets available we will be able to support certain scenarios often found in the software development process:

1. **Synthesis Scenarios**—support the development of new source code:
  - A developer could effectively query colleagues for support (expert finding) and/or being suggested contextualised code fragment(s);
  - A project manager could learn from previous projects and/or metrics.
2. **Analysis Scenarios**—support the project management and maintenance of existing source code:
  - One could perform opinion mining on SIOC [8] based representations of the discussion forums [19] in order to generate reports on a component or extract features requests and/or bug reports. SIOC (Semantically Interlinked Online Communities) is

a vocabulary to describe the content and structure of online community sites. It allows to create new connections between discussion channels and posts;

- Given that the documentation is interlinked with the source code, a dynamic FAQs could be provided;
- Developer profiles, based on their commitments to versioning control systems and the source code could be provided.

The contribution of this work is twofold: first, we introduce *Linked Data Driven Software Development* (LD2SD), a light-weight Semantic Web methodology to turn software artefacts such as data from version control systems, bug tracking tools or Java source code into linked data. Further, we show the application of LD2SD on a reference software project.

The paper is structured as follows: in section 2 we present use cases for LD2SD. Then, we introduce the overall LD2SD methodology in section 3 and discuss its characteristics in section 4. We report on exemplary implementation of LD2SD in section 5. In section 6 we review related work and compare it to our approach. Finally, we conclude in section 7 and outline future steps.

## 2. USE CASES

As motivated above, there are plenty of real-world scenarios one could think of where explicit interlinks between data sources would be desirable. We have detailed out a couple of use cases in the following which we realise in the realm of the software development. The use cases described below have in common that at least two data sets are involved. Note, that in case only *one* data set (such as bug tracking) is targeted, various solutions (cf. section 6) already exist, potentially not justifying the effort to apply LD2SD.

**Finding an expert** Jim has a long career in software project management. He knows that a task will be solved fast and bug-free only by a developer who is an expert in the required field. Jim now wants to assign a new task which involves Web pages scrapping. He needs to find a member of his team who is an expert in the HTML-Parser Java library.

**Issues not fixed in due time** Bug tracking systems contain a lot of *issue* entries. These *issues* need to be *fixed* on assigned dates. Harry, a project leader, is very busy, having to travel most of his time. He just came back from a project review and wants to know if all the issues due yesterday have been fixed. Harry additionally wants to know about the breakdown in terms of lines-of-code committed and which packages have been effected.

**Find developer replacement** Mary, a developer for a software company, has to relocate with her husband in another city. Julie, her supervisor, needs to hire a developer who can replace Mary. Therefore, Julie wants an analysis of her expertise and latest activities: assigned bugs, committed code, mailing list and blog posts, subsequently finding CVs that match Mary's expertise.

**Assigning a bug to a developer** Bug tracking environments structure *bugs* assignment by projects. John, a user of project *X* finds a bug and reports it on a *blog* post. Sarah, a developer of project *Y*, reads the *blog* post. However, she does not know the project *X* developers and their experience. She needs to find the most active developer in project *X* and assign the *bug* to him/her.

## 3. LD2SD FOUNDATIONS

We first introduce linked data, the foundation of *Linked Data Driven Software Development* (LD2SD), and then give an account of the LDSD methodology.

### 3.1 Linked Data

The basic idea of linked data [6] has first been outlined by Tim Berners-Lee in 2006<sup>2</sup>, where he described the linked data principles as follows: (i) all items should be identified using *URIs* [18] and these URIs should be *dereferenceable*, that is, using HTTP URIs allows looking up the an item identified through the URI, further (ii) when looking up an URI (an RDF [13] property is interpreted as a hyperlink), it leads to more data, and (iii) links to URIs in other datasets should be included in order to enable the discovery of more data. In contrast to the full-fledged Semantic Web vision, linked data is mainly about publishing structured data in RDF using URIs rather than focusing on the ontological level or inferencing. This simplification—comparable to what the World Wide Web did for hypertext—fosters a wide-spread adoption [4].

### 3.2 Methodology

In order to provide a *uniform* and *central* access to the different datasets, one needs to interlink, integrate and align them. Various techniques could potentially be utilised (see also section 6), however, given the arguments regarding linked data above, we decided to realise a linked-data driven approach. In Fig. 2 the overall LD2SD methodology is depicted. This methodology basically covers the layers as described in the following:

1. Assign URIs to all entities in software artefacts and convert to RDF representations based on the linked data principles, yielding LD2SD datasets;
2. Use semantic indexer, such as Sindice [15] to index the LD2SD datasets;
3. Use semantic pipes, such as the DERI Pipes (cf. section 5.2) allowing to integrate, align and filter the LD2SD datasets;
4. Deliver the information to end-users integrated in their preferred environment, such as discussed in section 5.3.

## 4. LD2SD CHARACTERISTICS

### 4.1 Scale to the Web

In 2007, the Linking Open Data (LOD) project<sup>3</sup>, an open, collaborative effort aiming at bootstrapping the Web of Data

<sup>2</sup><http://www.w3.org/DesignIssues/LinkedData.html>

<sup>3</sup><http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

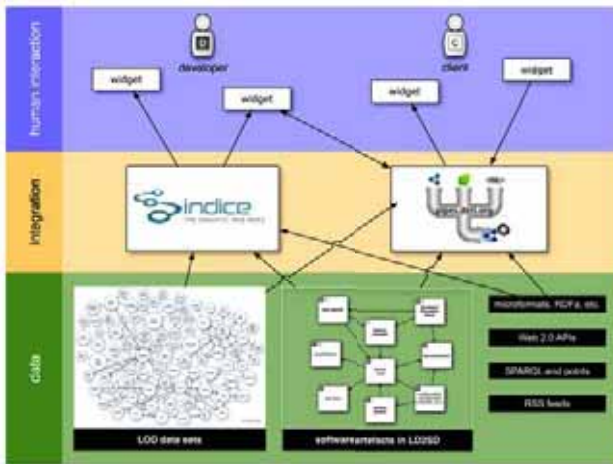


Figure 2: LD2SD Methodology.

by publishing datasets in RDF on the Web and creating interlinks between these datasets, has been launched. With over 50 interlinked dataset offering billions of RDF triples and millions of interlinks, the so called “LOD cloud” enables entire new application areas [11].

We highlight the fact, that by utilising LD2SD, a Web-scale data integration of software development-related information is hence made possible. One can imagine that—as both LD2SD and LOD follow the linked data principles—we are able to connect software artefacts to the LOD datasets, such as DBpedia [3], hence enabling the reuse of existing information in the software development process.

## 4.2 Read-only?

To this end, LD2SD allows us to integrate, view and filter the data. However, one problem remains unresolved: updating the original software artefact. With a recently launched community project called pushback<sup>4</sup>—aiming at turning the current “read-only” Semantic Web into a read/write Semantic Web—we are confident to adequately address this issue in the near future.

## 5. IMPLEMENTATION

The methodology described in the previous section will be demonstrated herein. We have divided this section according to Fig. 2: in section 5.1, we describe the reference data set and the interlinking, in section 5.2, we show the querying of the interconnected datasets using DERI Pipes<sup>5</sup>, and in section 5.3 we present some earlier work on Semantic Widgets.

### 5.1 Data Layer

We first present a list of candidate software artefacts to be converted to RDF and then we present a concrete example to realise some use case described in section 2.

To practice what we preach, we have chosen the Sindice software project<sup>6</sup> as the reference software project. In Table 1 the details regarding the respective LD2SD datasets

<sup>4</sup><http://esw.w3.org/topic/PushBackDataToLegacySources>

<sup>5</sup><http://pipes.deri.org/>

<sup>6</sup><http://sindice.com/>

are listed, yielding more than 43,000 (43k) RDF triples in total.

In order to apply the interlinking approach to our software artefacts as listed in the previous section we examine the RDF datasets in the following. An excerpt of an exemplary RDF representation of some Sindice Java source code is shown in listing 1. Further, an example of some

```

1 @prefix b: <http://baetle.googlecode.com/svn/ns/#> .
2 @prefix : <urn:java:org.sindice.projects.wp.> .
3 :WPLinkExtractor a b:Class;
4 b:contained <> ;
5 b:uses <urn:java:java.awt.Component> ,
6       <urn:java:java.io.IOException> .

```

Listing 1: An exemplary Java RDFication.

RDFised Subversion logs is shown in listing 2. From the

```

1 @prefix b: <http://baetle.googlecode.com/svn/ns/#> .
2 @prefix : <svn://sindice.com/svn/> .
3 :bc275 a b:Committing ;
4 b:added <svn://sindice/wp/WPLinkExtractor.java> .
5 b:author :oanure .

```

Listing 2: An exemplary Subversion RDFication.

listings 1 and 2, we are able to conclude that both RDF fragments are describing the same entity, “WPLinkExtractor.java”. We can interlink<sup>7</sup> these two RDF fragments as shown in listing 3 using an owl:sameAs property indicating that these URIs actually refer to the same entity.

```

1 :WPLinkExtractor owl:sameAs
2 <svn://sindice/wp/WPLinkExtractor.java> .

```

Listing 3: An Interlinking Example.

### 5.2 Integration Layer

After RDFising and interlinking the software artefacts, the next step is integrating the artefacts and query them.

DERI Pipes [16] are an open source project used to build RDF-based mashups. They allow to fetch RDF documents from different sources (referenced via URIs), merge them and operate on them. In our case at hand, this involves four major steps:

1. Fetch the RDF representation of the Subversion log, JIRA<sup>8</sup> issue tracker, Java source code, etc. using the *RDF Fetch operator*<sup>9</sup>;
2. Merge the datasets using a *Simple Mix operator*<sup>10</sup>;
3. Query the resulting, integrated dataset with SPARQL<sup>11</sup>;
4. Apply XQuery<sup>12</sup> in order to sort and format the data from the previous step.

<sup>7</sup><http://www.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/#RDFlinks/>

<sup>8</sup><http://www.atlassian.com/software/jira/>

<sup>9</sup><http://pipes.deri.org:8080/pipes/doc/#FETCH>

<sup>10</sup><http://pipes.deri.org:8080/pipes/doc/#MIX>

<sup>11</sup><http://www.w3.org/TR/rdf-sparql-query/>

<sup>12</sup><http://www.w3.org/TR/xquery/>

Software Artefact	Data Format	RDFizer	Vocabulary	Triples
JIRA Bug Tracker	relational data	D2RQ [7]	BAETLE <sup>a</sup>	12k
Java Source Code	structured data	SIMILE RDFizer <sup>b</sup>	SIMILE Java2RDF	22k
Subversion	relational data	BAETLE RDFizer <sup>c</sup>	BAETLE	7k
Developer's calendar	RFC2445 [10]	iCalendar to RDF <sup>d</sup>	iCalendar <sup>e</sup>	1k
Developer's profile	FOAF/RDF	system specific	FOAF <sup>f</sup>	1k
Developer blog	relational data	SIOC exporter <sup>g</sup>	SIOC [8] <sup>h</sup>	not yet implemented
Project Mailing Lists	RFC2822 [17]	SIMILE RDFizer <sup>i</sup>	SIMILE Email2RDF	not yet implemented

Table 1: The Sindice Reference Software Project.

<sup>a</sup><http://baetle.googlecode.com/svn/ns/>  
<sup>b</sup><http://simile.mit.edu/repository/RDFizers/java2rdf/>  
<sup>c</sup><http://code.google.com/p/baetle/>  
<sup>d</sup><http://www.kanzaki.com/courier/ical2rdf>  
<sup>e</sup><http://www.w3.org/TR/rdfcal/>  
<sup>f</sup><http://xmlns.com/foaf/spec/>  
<sup>g</sup><http://sioc-project.org/wordpress/>  
<sup>h</sup><http://rdfs.org/sioc/spec/>  
<sup>i</sup><http://simile.mit.edu/repository/RDFizers/email2rdf/>

The output of the implemented pipe is then accessible via an URI.

Let's consider the situation described in the *Issues not fixed in due time* use case (cf. section 2). The information we need to process is contained in a JIRA RDF dump<sup>13</sup> describing the issues assigned to a developer, and in the developers FOAF<sup>14</sup> files. The state of an issue can be *Open*, *Closed* or *Resolved*. We are interested in issues that are *Open* and that were due yesterday. Further, we want to display the issue summary and the author full name. In order to retrieve the information we are interested in, we apply a SPARQL SELECT query (listing 4).

```

1 PREFIX b: <http://baetle.googlecode.com/svn/ns/#> .
2 PREFIX w3s: <http://www.w3.org/2005/01/wf/flow#> .
3 SELECT ?issue ?author ?summary ?due_date
4 WHERE {
5   ?issue b:assigned_to ?author ;
6         b:due_date ?due_date ;
7         b:summary ?summary ;
8         w3s:state
9           <http://ld2sd.derI.org/data/Bugs2RDF/Open>
10        .
11 }

```

Listing 4: SPARQL Query to Select Overdue Issues.

As a matter of fact, SPARQL is currently limited to filter only specific dates. However, XQuery allows some basic calculations on top of the resulting SPARQL XML file, as shown in listing 5. In the XQuery box (see Fig. 4) we can specify the `Content-type:xml/html`, allowing us to format the output using HTML and directly display the result in a Web browser. The code snippet from listing 5 calculates the yesterday's date by subtracting one day from the current date (lines 4-5) and renders the *summary*, *author* and *issue* elements as rows in an HTML table (Fig. 3). In a second step, the developer's profiles exposed as FOAF can be integrated. This would for example mean that the URIs in the *Author*-column in Fig. 3 would be replaced by the

<sup>13</sup><http://ld2sd.derI.org/data/Bugs2RDF/RDFDump.rdf>

<sup>14</sup><http://www.foaf-project.org/>

```

1 for $b in .//*:result
2   where xs:date(xs:dateTime($b/*:binding[@name =
3     "due_date"]/*:literal))
4     = xs:date(xs:date(current-date())-
5       xs:dayTimeDuration("P1DT0H0M"))
6   return
7     <tr style="background: #CBE9C7; color:
8       black;">
9       <td> { $b/*:binding[@name =
10         "summary"]/*:literal } </td>
11       <td> { $b/*:binding[@name =
12         "author"]/*:uri } </td>
13       <td> { $b/*:binding[@name =
14         "issue"]/*:uri } </td>
15     </tr>

```

Listing 5: XQuery Filtering by Yesterday's Date.

Summary	Author	Issue URI
Add documentation generation to Maven build	<a href="http://www.derI.org:8080/LD2SD/SVN/gareth">http://www.derI.org:8080/LD2SD/SVN/gareth</a>	<a href="http://www.derI.org:8080/browse/SINFRA-24">http://www.derI.org:8080/browse/SINFRA-24</a>
Delete documents based on rules	<a href="http://www.derI.org:8080/LD2SD/SVN/michael">http://www.derI.org:8080/LD2SD/SVN/michael</a>	<a href="http://www.derI.org:8080/browse/SMDL-28">http://www.derI.org:8080/browse/SMDL-28</a>
IssueReader does not count correctly the number of items	<a href="http://www.derI.org:8080/LD2SD/SVN/andrea">http://www.derI.org:8080/LD2SD/SVN/andrea</a>	<a href="http://www.derI.org:8080/browse/SMDL-1">http://www.derI.org:8080/browse/SMDL-1</a>

Figure 3: Pipe Result in a Web Browser.

respective developer's full name. Further, by integrating the developer's profile data, one can be group developer by team-membership (for example "core", "API", etc.) or render dependencies on other developers.

In the same manner the data from Subversion can be integrated in a further step in order to enable the breakdown in terms of lines-of-code committed or the highlight which packages have been effected by a certain bug-fix. Concluding, the more data sets are integrated, the richer the queries may be.

A screen-shot of a pipe implementing the above example is depicted in Fig. 4.

### 5.3 Interaction Layer

The interaction layer handles the interaction between the integrated data as described above and the end-users, such as developers. We have shown elsewhere [21] how to utilise

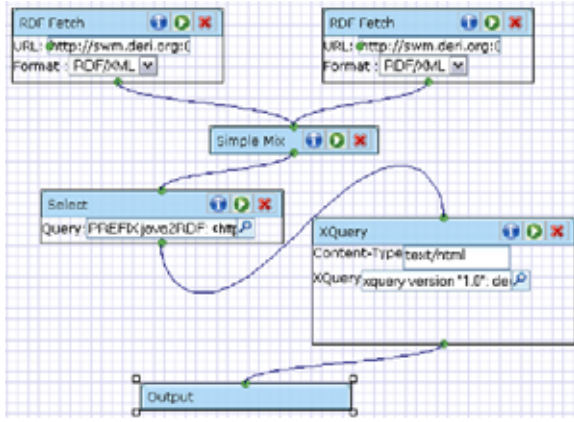


Figure 4: DERI Pipes.

the datasets using Semantic Widgets. With Semantic Widgets, we provide a methodology to enhance existing Web applications and deliver aggregated views of information to end-users. These views are accessed by clicking buttons



Figure 5: Examples of Semantic Widgets.

which are injected into the DOM of a Web page. For example, next to a bug, related information regarding bugs or dependent bugs is displayed as shown in Fig. 5.

## 6. RELATED WORK

There are certain technologies in the open source community available to combine software artefacts. Existing work related to combining software artefacts has been described in Dhruv by Ankolekar et.al. [1], a Semantic Web system for open source software (OSS) communities. It provides a semantic interface allowing users to see a detailed description of highlighted terms in the message posted during bug resolution in cross-links pages. Their approach extracts information about software artefacts using information extraction techniques based on noun phrases, code terms and references to other artefacts. Dhruv is specifically designed for OSS bug resolution processes.

Another interesting, closely related work has been described in [2]. There, a relational database is used to store information related to version control and bug tracking data. The source code meta model has been represented using the Rigi Standard Format (RSF) [20], which is a triple based specification language that can be easily customised [2]. The integration of these three artefacts has been done by (i)

querying the relational database, and (ii) merging the result with the source model RSF files. In contrast to their approach, we have provided a methodology to integrate the candidate software artefacts by RDFizing and interlinking them using linked-data driven approach.

In [12], Kiefer et.al. have presented EvoOnt<sup>15</sup>, a software repository data exchange format based on OWL<sup>16</sup>. EvoOnt includes software code, code repository and bug information. They have used the iSPARQL<sup>17</sup> engine which is an extension of SPARQL, to query for similar software entities. iSPARQL is based on *virtual triples* which are used to configure *similarity joins* [9]. Their approach differs from our approach in that we have used DERI pipes to integrate and query different software artefacts.

Existing work related to our use cases discussed in section 2 has been described in [14]. Their approach uses data from change management systems and heuristics are based on but are limited to only two software artefacts, i.e., for software bugs and source code commits. Contrary to their approach, we have added developer’s information from blogs, mailing lists and developer’s profile to realize our use cases.

In [5], Basili et.al. have presented Experience Factory concept for software development. Experience Factory supports the evolution of processes and other forms of knowledge, based on experiences within the organization [5]. Experiences are captured from FAQs, chat logs, emails and project presentations. In contrast to their approach, we have provided a methodology to capture knowledge from candidate software artefacts and interlink them to find a certain expertise.

Mylyn<sup>18</sup> is a sub-system for the Eclipse IDE allowing multitasking for developer and task management. It provides the means render bug-related data in the Eclipse IDE for developers to work efficiently in their development environment without having to log in to the Web based application to update or create new bugs. The limitation of Mylyn is that it works only with task repositories such as JIRA, Bugzilla<sup>19</sup>.

Further, there are plug-ins<sup>20</sup> which integrates Subversion with bug trackers, for example Bugzilla or JIRA. The plug-in displays all Subversion commit messages related to a specific issue. To the best of our knowledge such Subversion plug-ins are available for a few bug trackers.

Existing work described above somehow try to address the integration or interlinking of different software artefacts but some of them are desktop applications and some are Web based applications and none of the above described approaches address all the candidate software artefacts we described in this paper (see Table 1). Still what is missing is the existence of a generic framework where all software artefacts can be collected and queried that would allow project managers to get an overall view of the software development projects.

<sup>15</sup><http://www.ifi.uzh.ch/ddis/evo/>

<sup>16</sup><http://www.w3.org/TR/2004/REC-owl-guide-20040210/>

<sup>17</sup><http://www.ifi.uzh.ch/ddis/isparql.html>

<sup>18</sup><http://www.eclipse.org/mylyn/>

<sup>19</sup><http://www.bugzilla.org/>

<sup>20</sup><http://subversion.tigris.org/links.html#misc-utils>

## 7. CONCLUSION & OUTLOOK

We have motivated and introduced Linked Data Driven Software Development (LD2SD) as well as demonstrated its value in a concrete setup in this paper. The basic idea underlying LD2SD is to make the **implicit links between software artefacts** found in software development—such as version control systems, issue trackers, discussion forums, etc.—**explicit** and expose them using RDF. By using LD2SD, one enables a Web-scale integration of data, connecting to the LOD cloud and enabling the vast reuse of information.

We plan to implement further LD2SD use cases to show how one can benefit from it, especially when more than two data sources are involved. We aim to overcome a shortcoming of the current implementation, as it is not 100% linked data conforming; in certain places we use URNs rather than HTTP URIs. Additionally we will improve the interlinking, yielding higher-quality and also more links between the LD2SD datasets.

## Acknowledgements

Our work has partly been supported by the European Commission under Grant No. 217031, FP7/ICT-2007.1.2, project Romulus—“Domain Driven Design and Mashup Oriented Development based on Open Source Java Metaframework for Pragmatic, Reliable and Secure Web Development”<sup>21</sup>.

## 8. REFERENCES

- [1] A. Ankolekar, K. Sycara, J. Herbsleb, R. Kraut, and C. Welty. Supporting online problem-solving communities with the Semantic Web. In *Proceedings of the 15th International Conference on World Wide Web*, Edinburgh, Scotland, 2006.
- [2] G. Antonio, M. D. Penta, H. Gall, and M. Pinzger. Towards the Integration of Versioning Systems, Bug Reports and Source Code Meta-Models. In *Electronic Notes in Theoretical Computer Science*, pages 87–99, 2005.
- [3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, pages 722–735, 2007.
- [4] D. Ayers. Evolving the Link. *IEEE Internet Computing*, 11(3):94–96, 2007.
- [5] V. R. Basili, M. Lindvall, and P. Costa. Implementing the Experience Factory concepts as a set of Experience Bases. In *Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering*, pages 102–109, 2001.
- [6] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked Data on the Web (LDOW2008). In *Linked Data on the Web Workshop (WWW2008)*, 2008.
- [7] C. Bizer and A. Seaborne. D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs. In *3rd International Semantic Web Conference*, Hiroshima, Japan, 2004.
- [8] U. Bojars, J. Breslin, V. Peristeras, G. Tummarello, and S. Decker. Interlinking the Social Web with Semantics. In *IEEE Intelligent Systems*, 23(3):29–40, 2008.
- [9] W. W. Cohen. Data Integration Using Similarity Joins and a Word-Based Information Representation Language. In *ACM TOIS*, pages 288–321, 2000.
- [10] F. Dawson and D. Stenerson. Internet Calendaring and Scheduling Core Object Specification (iCalendar), RFC2445. IETF Network Working Group, 1998. <http://www.ietf.org/rfc/rfc2445.txt>.
- [11] M. Hausenblas. Exploiting Linked Data For Building Web Applications. *IEEE Internet Computing*, N(N):to appear, 2009.
- [12] C. Kiefer, A. Bernstein, and J. Tappolet. Mining Software Repositories with iSPARQL and a Software Evolution Ontology. In *Proceedings of the ICSE International Workshop on Mining Software Repositories (MSR)*, Minneapolis, MA, 2007.
- [13] G. Klyne, J. J. Carroll, and B. McBride. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004, RDF Core Working Group, 2004.
- [14] A. Mockus and J. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. In *Proceedings of the 24th International Conference on Software Engineering, Orlando*, 2002.
- [15] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1):37–52, 2008.
- [16] D. L. Phouc, A. Polleres, C. Morbidoni, M. Hauswirth, and G. Tummarello. Rapid Prototyping of Semantic Mash-Ups through Semantic Web Pipes. In *Proceedings of the 18th International World Wide Web Conference (WWW2009)*, ACM, Madrid, Spain, 2009.
- [17] P. Resnick. Internet Message Format, RFC2822. IETF Network Working Group, 2001. <http://www.ietf.org/rfc/rfc2822.txt>.
- [18] L. Sauermann and R. Cyganiak. Cool URIs for the Semantic Web. W3C Interest Group Note 31 March 2008, W3C Semantic Web Education and Outreach Interest Group, 2008.
- [19] S. Softic and M. Hausenblas. Towards Opinion Mining Through Tracing Discussions on the Web. In *Social Data on the Web (SDoW 2008) Workshop at the 7th International Semantic Web Conference*, Karlsruhe, Germany, 2008.
- [20] S. R. Tilley, K. Wong, M. A. D. Storey, and H. A. Muller. Programmable Reverse Engineering. In *International Journal of Software Engineering and Knowledge Engineering*, pages 501–520, 1994.
- [21] A. Westerski, A. Iqbal, G. Tummarello, and S. Decker. Sindice Widgets: Lightweight embedding of Semantic Web capabilities into existing user applications. In *Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering, ISWC08*, 2008.

<sup>21</sup><http://www.ict-romulus.eu/>

# Improving Searchability of a Music Digital Library with Semantic Web Technologies

Paloma de Juan  
Departamento de Ingeniería  
de Sistemas Telemáticos  
Universidad Politécnica de Madrid  
Email: paloko@gsi.dit.upm.es

Carlos Á. Iglesias  
Germinus XXI (Grupo Gesfor)  
Email: cif@germinus.com

**Abstract**—Traditional search systems are usually based on keywords, a very simple and convenient mechanism to express a need for information. This is the most extended way of searching the Web, although it is not always an easy task to accurately summarize a natural language query in a few keywords. Working with keywords means losing the context, which is the only thing that can help us deal with ambiguity. This is the biggest problem of keyword-based systems. Semantic Web technologies seem a perfect solution to this problem, since they make it possible to represent the semantics of a given domain. In this paper, we present three projects, Harmos, Semusici and Cantiga, whose aim is providing access to a music digital library. We will describe two search systems, a traditional one and a semantic one, developed in the context of these projects and compare them in terms of usability and effectiveness.

## I. INTRODUCTION

For some years now, we have been living in a world where the Web has been dominated by plain textual contents. These have been reachable thanks to search engines and directories, which have been designed to work in a keyword environment. The main problem of a keyword-based system is ambiguity. Unfortunately, the meaning of a keyword can only be determined by its surroundings (if available). The concept of “context” can not be applied in this situation.

The same happens when we look for multimedia resources, which are becoming more and more important lately. A picture (or a video or audio file) can not usually be reduced to a set of words<sup>1</sup>. In order for users to share this kind of contents, they must provide some keywords to make them reachable by other users. In this way, a conventional system can give access to both textual and multimedia resources. There are hundreds of relationships between the semantic descriptors used to tag the multimedia resources. However, this information is not taken into account when a search is processed.

The type of queries a keyword-based system can accept are quite limited and semantically poor. A keyword in a text field can mean anything. We have no information about its nature: it could be the name of a city, an address, the title of a book, a date, a person’s name or even an identifier. If we named that text field, we could partially restrict the meaning of the keyword, e.g. the keyword is a date. But what is the semantics

<sup>1</sup>It is possible to automatically extract features from multimedia resources and use them as tags, but it is a costly process.

of this date? In the context of the projects this paper presents, it could be the date a master class was recorded, the date a composition was composed, the birth date of a composer... We need to move to a new search paradigm that allows us to ask more semantically complex questions, e.g. “I want to find compositions by Nordic composers.”

Changing the search paradigm or, let us say, the system interface, would not be enough to provide better results. We could find a way to let the user express very accurately what she is looking for but we will need to change the structure that supports the knowledge of the system if we really want to exploit the semantic relationships that link all the concepts involved. An ontology may help us define rules that would enrich the knowledge base with more information than what is explicitly stated.

In this paper, we will present the changes we have introduced in a traditional system in order to improve its searchability both in terms of usability (changing the interface) and effectiveness (changing the structure that supports the knowledge base). We will also discuss how we have progressively improve our system in the context of three projects we have been or are involved in: Harmos, Semusici and Cantiga.

In Section II, we will present our previous and current work in the field, describing the three projects we have just introduced. In Section III, we will describe MagisterMusicae, the system built for Harmos. In Section IV, we will explain how we have improved this system in the context of Semusici and Cantiga. In Section V, we will describe Cantiga Search System. An evaluation of the improvements of this system over the one introduced in Section III will be conducted in Section VI. Finally, we will review some related projects and present our conclusions and future work in Sections VII and VIII, respectively.

## II. RESEARCH CONTEXT

### A. Previous Work

The work presented in this article comes from the experience in several projects related to music digital libraries with the common aim of providing Internet access to music resources hold by different institutions. The collection we have been working with contains more than 700 audiovisual hours

of recorded master classes, property of Fundación Albéniz [1]. These resources have been tagged according to a set of tags that define a vocabulary of pedagogical concepts, which we will talk about later.

The first of this series of music related projects we have been involved in was European eContent Harnos project. Harnos produced a collection of audiovisual contents belonging to the music heritage, where education was the principal focus and the project's main objective. The resulting system is available at <http://www.magistermusicae.com>. We will describe this system in the next section. The aim of the second project, Semusici, was to improve Harnos system by applying Semantic Web technologies. The main output of this project was Semusici ontology.

### B. Cantiga Project

The goal of the last of this series of projects, Cantiga, is to investigate how Web 2.0 technologies can be applied to the cataloging and search of music resources. In this project, we are trying to develop a platform that will help users annotate and find contents of digital libraries from different music institutions. This platform is also intended to provide a framework to help these institutions communicate (both internally and externally) and interact, allowing them to create workflows. These workflows should help automating such tasks as translation, quality control and other administrative procedures. The system will also support federated search across all the libraries available.

In order to decrease the high cost of manual cataloging, the system uses advanced tools to semi-automatically extract features from the multimedia resources. This will help users annotate these resources, which will make it easier to retrieve them. We are also using Semusici ontology to classify the resources. Using a formal structure to model the domain will make searching faster and will increase the precision and recall of the system. The reason is that users will be able to produce more accurate queries and will get a whole set of semantically related results.

## III. MAGISTERMUSICAE SEARCH SYSTEM

MagisterMusicae, the system developed in the context of Harnos, is a simple search system based on facets. This is a search paradigm in which keywords are placed in slots<sup>2</sup>, the so-called *facets*, that have a certain semantics. The main interface consists on a series of drop down menus (as shown in Fig. 1) that allow the user to search a master class given the instrument it is oriented to, the teacher giving the lecture and the composer who composed the piece of music being played in the class. To simplify the process of searching, a teacher can only be selected in case an instrument has already been chosen. Also, a teacher has to be selected before choosing a composer.

The advanced search interface (Fig. 2) allows the user to select a series of keywords

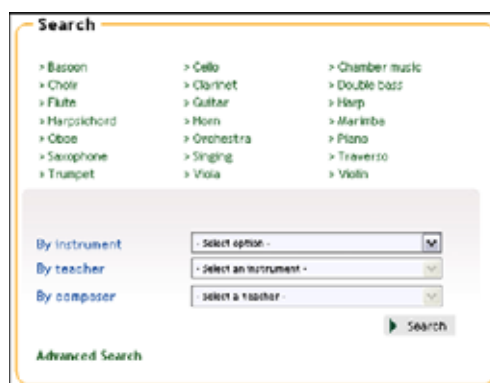


Fig. 1. MagisterMusicae Basic Search Interface

belonging to seven different categories: Instruments, Teachers, Students, Concepts, Composers, Works and Movements. This makes it possible to look for a master class without knowing anything about the instrument it is addressed to. There is much more information about the master classes available in the knowledge base (dates, places, languages...) that can not be used to build a query in this system. Including a tab for every single category would probably have made users refuse the system.



Fig. 2. MagisterMusicae Advanced Search Interface

Neither of the two search systems allow the user to type any term of her choice, as they both provide a guided search service. Internally, MagisterMusicae uses a relational database to store the information (no ontology is used). The selected keywords are used to build a simple SQL query which is expected to deliver a list of master classes fulfilling all the requirements the user has posed.

## IV. IMPROVING SEARCHABILITY

The system we have just described has certain limitations, which will be discussed in the next section. The following subsections will present our work in improving the way users access the contents of the Fundación Albéniz master classes collection. We have been particularly concerned about the effectiveness of the system in terms of number of results delivered when the knowledge base has no explicit information about some fact. This is a key point in Semantic Web technologies, since the structures they propose have the ability to make explicit knowledge that is implicit in a given domain [2].

<sup>2</sup>In this case, the user is asked to select values instead of filling text fields.



### A. Applying Semantic Web Technologies

As we have already said, Semusici intended to improve the results of Harnos by introducing Semantic Web technologies. An ontology was built in order to support all the information contained in Harnos database. As we will see later in this section, most of the knowledge is hidden behind the relationships between the concepts involved. An ontology, by definition, represents a formal model of the common interpretation of the entities and relationships of a given domain (or, as Gruber said, “an explicit specification of a conceptualization” [3]). Therefore, it has great powerful retrieval and inferential capabilities. Another reason why we chose this structure is that it makes it possible to easily import information from external sources in order to enrich the knowledge base.

The knowledge base we have been working on has two distinct parts. The first one captures all the information that can be useful to answer any query that is not directly related to the pedagogical aspects of a master class. For instance, “Show me all the recordings related to composers born in the 18th century.” This part of the knowledge base contains all the information about the context of a master class, mainly geographical and bibliographical data.

The other part of the knowledge base is the concepts taxonomy. This taxonomy contains over 350 pedagogical concepts that are used as tags to describe the recordings. It was built from Harnos pedagogical taxonomy [4] following a bottom-up strategy in order to redistribute the concepts in a way their relationships could be exploited<sup>3</sup>. This taxonomy aims to cover the whole spectrum of music practice and teaching, focusing on pedagogical aspects, such as technique (general or specific of an instrument), mechanics, musicology, musical elements (rhythm, melody, harmony, form...), etc.

Semusici ontology consists of more than 150 classes and almost 40 properties. We may distinguish four substructures:

- A Domain ontology. It includes classes such as *Composition*, *Instrument*, *Composer*, *Teacher*, and other concepts that characterize a composition, such as *Genre*, *Style* or *Form*. It also contains the class *Person*, representing those who are involved in a master class or have composed a piece of music, and the class *Place*, which will help us geographically locate the master classes, as well as indicate where a person was born or dead.
- The Instrument taxonomy, which is part of the domain ontology. Instruments have been classified according to the family they belong to. Every instrument is an instance of a class representing its family (*StringInstrument*, *WindInstrument*, *PercussionInstrument*...). The taxonomy these families conform has been modelled using SKOS [5].
- A Resources ontology. This ontology models the multimedia resources that support the master classes

<sup>3</sup>The original distribution had very little semantic information and the elements in each level were not always equally specific.

(mostly videos, but also audio files and documents) and their features. It includes concepts such as *Class*, *Multimedia* and its subclasses *Video*, *Audio* and *Document*, and the properties *title*, *date*, *language*, *targetAudience*, etc.

- The Concept taxonomy. This taxonomy contains the new and improved distribution of the pedagogical concepts assigned to the master classes and has also been modelled using SKOS. The different categories are arranged hierarchically under the class *Concept*. Properties such as *relatedTo*, *partOf* and *elementOf* are used to semantically relate concepts.

### B. Linking the Ontology with External Data Sources

We also decided to include links to external data sources. Our aim was to populate our ontology with information that is not usually provided by the annotators, but is related to the subject of the master classes. There are many sites that offer a wide variety of RDF data sources, like Geonames [6], MusicBrainz [7], CIA Factbook [8], DBpedia [9], etc.

We wanted to provide a way to allow the system perform geographical entailments. We chose CIA Factbook as our source. The CIA Factbook is an annual publication of the Central Intelligence Agency of the United States with information about the countries of the world. The Factbook provides a summary of the demographics, geography, communications, government, economy, and military of 266 countries, dependencies and other areas in the world.

We linked our geographical resources (instances of the class *Place*) with the corresponding entries in the CIA Factbook. This means we can now relate composers, compositions and master classes in terms of their location. We could even geolocate them and draw a map with all the items associated to each place, in order to help users find information in a more visual way. Moreover, this newly incorporated knowledge can help us find resources in an area of any size, even if the only information we have is the name of a city somehow associated to those resources (being the place where the class took place, the place where a composer was born...).

### C. Alternative Search Paradigms

The purpose of the Semantic Web is to improve the way users access the vast amount of information that is available through the Internet. The systems providing this access have made users change their natural way of expressing their need for information, that is using natural language. At this moment, the use of keyword-based queries is so extended that is difficult to conceive any other mechanism of searching the Web. For most people, it is very simple and fast to summarize what is in their heads in a few words (which is actually very little information about what they are looking for).

The main problem of keyword-based systems is ambiguity. The correct meaning of a word can not be determined without considering its context. Unfortunately, in a traditional keyword-based system there is no such context. The key to solve this is adding semantics both to the query and the

resources users are looking for. Of course this would mean to restructure the whole Web, which is impossible. But applying this to restricted domains can really improve the search.

There are many ways users can express their need for information without losing the context. A popular paradigm of search is faceted search. As we have already seen (as it is the case of *MagisterMusicae*), we build the context of the query by navigating through different categories, which are usually arranged in a taxonomy.

In order to fully keep the context of the query, users should express it in natural language. Unfortunately, it is very difficult for a system to correctly process a natural language query. We would then need a solution combining both the advantages of semantics and keywords. The nearest solution to a natural language processing (NLP) system would be a template-based one.

In a template-based system, we associate a template to a keyword (or a set of keywords). This makes it possible to produce more complex queries. For instance, we could specify that the date in the example we proposed in the Introduction is “the date when the composer of the composition that is referred to in the master class I am looking for was born.” The only thing the user has to do is select the template that best suits the semantics of the keyword she wants to search for.

The number of templates should be limited to a few ones in order not to overwhelm the users with too many alternatives. Otherwise, they may feel they are choosing an option among the available ones. Instead, we want to provide them with an intuitive way to build the request they have in mind. Users may be able to quickly choose the proper context to what they know about what they are looking for.

## V. CANTIGA SEMANTIC SEARCH SYSTEM

The prototype built in the context of Cantiga is a result of all the improvements presented in the last section. Its core is an extended version of *Semusici* ontology and its interface is based on templates. We analyzed the current state of the knowledge base and discarded those queries that would not retrieve any content. This dramatically decreased the number of possible templates. However, the underlying ontology allows a much more diverse set of queries, based on properties of compositions and composers that have not yet been used. This will make it possible to include a whole lot of new queries when the knowledge base grows.

We have adapted the traditional one-level model of templates into a hierarchical one. Instead of using a single level of templates, we decided to group the queries according to common components of meaning, in order to let the user combine these pieces and build a whole template. Our intention is to give her the impression of being progressively restricting the meaning of the piece of information she wants to search for. Besides, we do not want to overwhelm her by offering her too many options at a time, as we have already said.

We built a tree with the fragments of templates. This tree has up to 5 levels of depth. Each branch defines a restriction on the meaning of the piece of information the user is looking for. A total of 35 templates were defined, each of them represented by a path that goes from the parent node to each leaf node of the tree.

The parent node of this tree is “*Which classes,*” since that is what the user is ultimately looking for. The first level contains fragments of queries about the parameters of a master class. For example, we find the piece “*were held in X?*,” which means that the value “X” introduced by the user is not only a place, but “the place where the master classes took place.” In the case of “*are addressed to X?*,” “X” is “the audience for whom the classes are meant.”

Whenever we found a parameter representing a complex concept, e.g. a *Composition*, a new level was added to the tree. Following this example, we created a second level whose parent node is “*refer to a composition.*” This level contains new fragments of queries concerning the properties of a composition, e.g. “*of the form X,*” “*composed by,*” etc. We proceeded the same way until every path reached a leaf node, i.e. one that contained a field to be filled by the user.

This way, the user would build a template selecting the path that best restricts the meaning of the term she intends to look up, e.g. “*Which classes refer to a composition composed by someone born in X?*” The interface of this system can be seen in Fig. 3.

## VI. EVALUATION

Cantiga search interface proved to be much more easy-to-use and intuitive than *MagisterMusicae*’s. First, in *MagisterMusicae* the user was expected to select an instrument before continue searching, which is pretty convenient in case one is a performer or a music student. However, this is a huge drawback if you are just interested in master classes taught by a certain teacher or referring to a certain piece of music, no matter what instrument they are focused on.

Of course there is the advanced search interface, but still this is not the interface presented to the user in the first place. Neither of *MagisterMusicae* interfaces allows the user to provide any keyword of her own. She will need to find the piece of information she already knows among hundreds of options in order to select the proper value.

Cantiga search interface, on the other hand, provides a simple way to build a query that is really expressed in natural language. As opposed to *MagisterMusicae*’s case, the user will not be selecting search parameters but the context of a keyword she will be able to provide. In the worst case, the user will have to make five selections (which is the number of levels of the template tree) in order to complete a whole template. However, the feeling she will get is the feeling of building a sentence and not just adding conditions to a query. In short, Cantiga template system provides a natural way that feels closer to the way human beings express restrictions.

One thing that has not been considered in Cantiga is conjunctive queries. While *MagisterMusicae* advanced search

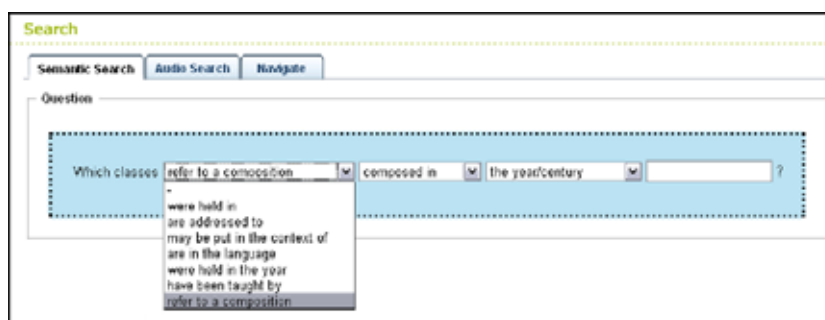


Fig. 3. Cantiga Semantic Search System

allows the user to look for a master class establishing more than one condition, Cantiga search system is only able to take one piece of information provided by the user at a time. This could be arranged by letting the user choose not just one template, but any number of them (one for each known detail about the master class she is looking for). For example, “Which classes refer to a composition composed by someone born in X and composed for an instrument of the Y family?”

About the coverage, we tested both systems in order to check if they met the users’ needs. We took 73 sample queries that were collected<sup>4</sup> during the specification phase of the ontology building process as a test set. We had previously used those queries as competency questions [10] in order to select the concepts the ontology was expected to formalize. This set included all kinds of questions, from rather simple ones (e.g. “I want to find classes taught by Argentinian teachers”) to very complex biographical ones (e.g. “Find all the classes referring to composers who used to work on commission”).

It turned out only 8 out of the 73 queries were considered in the first system, whereas 18 were included among the 35 templates of Cantiga search system. Even if we increased the number of facets, only 13 more queries could be processed by MagisterMusicae search system. This limitation is due to the lack of flexibility of the facet search paradigm, in terms of semantics. It is impossible to express a complex relationship in a system based on facets. For instances, we could never build a query such as “Which classes have been taught by a teacher whose first name is the same as Liszt’s?” or “Find all the classes referring to works composed by the author of ‘Tosca.’ ”

The combination of a semantic layer and a template-based interface is what makes Cantiga search system much more powerful. That is why up to 30 more of the test queries could be included as templates in this system. In fact, the reason why there are yet 25 more questions that could not be processed by this system is that they deal with rare concepts we decided not to include in the ontology.

Perhaps the greatest value of Cantiga search system lies in its expandability. Adding a new query to this system can be done by just adding the corresponding template, whereas

<sup>4</sup>A survey was carried out among musicians and music students and lovers in order to find out what kind of queries they would like to be able to ask.

adding a new query to MagisterMusicae’s involves not only including a new facet, but also showing every possible search value the user could introduce. And this would only be possible assuming the semantics of the query can be expressed using facets.

Finally, there is still another important reason why Cantiga search system performs better than MagisterMusicae. Let us say a user wants to look for master classes about strings technique. The knowledge base may have no record of any class related to the concept “strings technique,” yet the system would be able to retrieve some results concerning “violin technique” or “double bass technique.” The reason for this is that the ontology contains information that links these concepts. The mere fact of placing them in a hierarchy represents some implicit knowledge that can be used in situations such as this (i.e. a parent-child is inferred).

The interconnection with an external datasource such as the CIA Factbook also allows to search using all kind of geographical data. For instance, the system would provide an answer to “Which master classes have taken place in a European country?,” although such information is not present in our knowledge base. We could even find “master classes referring to a composer born in any country bordering Austria.”

## VII. RELATED WORK

In the past few years, there has been interesting research on the field of semantic search. In [11], the possibilities of using a view-based search paradigm to create intelligent search interfaces on the Semantic Web are explored. This thesis also presents a survey on semantic search related projects. Five research directions are identified: augmenting traditional keyword search with semantic techniques, basic concept location, complex constraint queries, problem solving and connecting path discovery. Our system would be part of the third group. According to this analysis, the main concern of this group is developing user interfaces that make creating complex query patterns as intuitive as possible. Other examples following this direction would be [12], [13] and [14].

There are some other approaches to template-based semantic search. In [15], a non-hierarchical template system is presented. This system uses templates of queries expressed in natural language with variable parts for substitution purposes.

These queries correspond to real-life questions and problems, just like in our case. The system was built into the JeromeDL [16] system, a semantic digital library engine that uses Semantic Web and Social Networking technologies to improve browsing and searching for resources.

This template-based system intends to provide access to publications, such as articles, books, etc., using only five templates. We have to consider that the domain it covers is much more limited than the one covered by Cantiga. The semantics of these templates is rather simple. Therefore, in this case it would not be necessary to split up the templates. Still, a flat structure such as this would not be acceptable if the number of templates increased. Another difference with our system is that it works with conjunctive queries, as one of the templates presents two slots to be filled by the user.

A more complex solution to semantic search is proposed in [17]. They present an approach for translating keyword queries to DL conjunctive queries using background knowledge available in ontologies, i.e. formally interpreting keyword queries. As we too did before, they discuss whether users really want to express themselves using natural language or maybe they find working with queries satisfying enough.

Finally, we can find some interesting web portals related to semantic search in the specific domain of music resources. The most important one is mSpace [18]. This service provides access to musical contents using bibliographical information associated to those contents, their classification and the relation between the corresponding categories. There are also some other interesting works on applying Semantic Web technologies to digital libraries, like [19] or [20].

### VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented our work in the field of semantic search through three projects whose purpose was to provide access to a music digital library. We have compared two different systems developed in the context of these projects in terms of usability and effectiveness. The semantic search system proved to be more flexible and powerful than the traditional one, thanks to the use of an ontology and a template-based interface.

We have proposed a solution to semantic search that combines both the advantages of semantics and keywords. Our hierarchical template-based prototype does not support user-generated natural language queries, but it includes a set of real-life questions that can be extended as needed. We will keep on researching new ways of searching that do not entail the drawbacks of a NLP system, but allow a more flexible and intuitive way of expressing the semantics of a query.

Finally, we have enriched our ontology by linking it to the CIA Factbook. We are currently working on linking it to other DBpedia datasets in order to improve the coverage of the system. We would also like to exploit a lexical resource such as WordNet [21] to perform semantic query expansion.

### ACKNOWLEDGMENTS

The research presented in this paper has been partially supported by the Spanish Ministry of Industry, Tourism and

Trade under the National Plan of R&D, in the context of Semusici (FIT-350200-2006-70 and FIT-350200-2007-44) and Cantiga (FIT-350201-2007-8) projects.

### REFERENCES

- [1] "Fundación Albéniz." [Online]. Available: <http://www.fundacionalbeniz.com>
- [2] M. Uschold and M. Grüninger, "Ontologies: Principles, Methods, and Applications," *Knowledge Engineering Review*, vol. 11, no. 2, pp. 93–155, 1996. [Online]. Available: <http://citeseer.ist.psu.edu/uschold96ontologie.html>
- [3] T. R. Gruber, "Towards Principles for the Design of Ontologies Used for Knowledge Sharing," in *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, Eds. Deventer, The Netherlands: Kluwer Academic Publishers, 1993. [Online]. Available: [citeseer.ist.psu.edu/gruber93toward.html](http://citeseer.ist.psu.edu/gruber93toward.html)
- [4] C. Á. Iglesias, M. Sánchez, Álvaro Guibert, M. J. Guibert, and E. Gómez, "A Multilingual Web based Educational System for Professional Musicians," *Current Developments in Assisted Education*, 2006.
- [5] "SKOS Simple Knowledge Organization System." [Online]. Available: <http://www.w3.org/2004/02/skos>
- [6] "GeoNames." [Online]. Available: <http://www.geonames.org>
- [7] A. Swartz, "MusicBrainz: A Semantic Web Service," 2002. [Online]. Available: [citeseer.ist.psu.edu/swartz02musicbrainz.html](http://citeseer.ist.psu.edu/swartz02musicbrainz.html)
- [8] "CIA - The World Factbook." [Online]. Available: <https://www.cia.gov/library/publications/the-world-factbook>
- [9] "DBpedia." [Online]. Available: <http://dbpedia.org>
- [10] M. Grüninger and M. S. Fox, "Methodology for the Design and Evaluation of Ontologies," in *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*, 1995. [Online]. Available: [citeseer.ist.psu.edu/grminger95methodology.html](http://citeseer.ist.psu.edu/grminger95methodology.html)
- [11] E. Mäkelä and T. Författare, "View-based Search Interfaces for the Semantic Web," Tech. Rep., 2006.
- [12] N. Athanasis, V. Christophides, and D. Kotzinos, "Generating on the fly queries for the semantic web: The ics-forth graphical rql interface (grql)," 2004, pp. 486–501. [Online]. Available: <http://www.springerlink.com/content/4mlgmjqwffga031k>
- [13] T. Catarci, P. Dongilli, T. D. Mascio, E. Franconi, G. Santucci, and S. Tessaris, "An Ontology Based Visual Tool for Query Formulation Support." Proc. of the 16th European Conference on Artificial Intelligence (ECAI'04), 2004, pp. 308–312.
- [14] L. Zhang, Y. Yu, Y. Yang, J. Zhou, and C. Lin, "An Enhanced Model for Searching in Semantic Portals," in *In WWW'05: Proceedings of the 14th international conference on World Wide Web*. ACM Press, 2005, pp. 453–462.
- [15] S. R. Kruk, K. Samp, C. O'Nuallain, B. Davis, and B. M. S. Grzonkowski, "Search Interface Based on Natural Language Query Templates." Proc. of the poster session of IADIS International Conference WWW/Internet 2006, 2006.
- [16] S. R. Kruk, T. Woroniecki, A. Gzella, and M. Dabrowski, "JeromeDL - a Semantic Digital Library," in *Semantic Web Challenge*, ser. CEUR Workshop Proceedings, J. Golbeck and P. Mika, Eds., vol. 295. CEUR-WS.org, 2007. [Online]. Available: <http://dblp.uni-trier.de/db/conf/semweb/challenge2007.html#KrukWGD07>
- [17] T. Tran, P. Cimiano, S. Rudolph, and R. Studer, "Ontology-Based Interpretation of Keywords for Semantic Search," in *ISWC/ASWC*, ser. Lecture Notes in Computer Science, vol. 4825. Springer, 2007, pp. 523–536.
- [18] "mSpace." [Online]. Available: <http://mspace.fm>
- [19] L. Hollink, A. Isaac, V. Malaisé, and G. Schreiber, "Semantic Web Opportunities for Digital Libraries." Proc. of 32nd Library Systems Seminar of the European Library Automation Group (ELAG), Wageningen, The Netherlands, 2008, 2008.
- [20] S. Kruk, T. Woroniecki, A. Gzella, M. Dabrowski, and B. McDaniel, "The anatomy of a Social Semantic Digital Library." Proc. of International Semantic Web Conference (ISWC), Athens, GA, USA, 2006, 2006.
- [21] "WordNet: A lexical database for the English language." [Online]. Available: <http://wordnet.princeton.edu>

# A Guideline Engine For Knowledge Management in Clinical Decision Support Systems (CDSSs)

Michele Ceccarelli<sup>a</sup>, Alessandro De Stasio<sup>a</sup>, Antonio Donatiello<sup>b</sup>, Dante Vitale<sup>b</sup>

<sup>a</sup> University Of Sannio, RCOST (Research Centre On Software Technology), Benevento, 82100 Italy

<sup>b</sup> Unlimited Software S.r.l., Napoli, 80143 Italy

e-mail: ceccarelli@unisannio.it, a.destasio@u-s.it, a.donatiello@u-s.it, dv@unlimitedsoftware.it

## Abstract

*The application of scientific methodology to clinical practice is typically realized through recommendations, policies and protocols represented as Clinical Practice Guidelines (CPGs). CPGs have the purpose to help the clinicians in their choices and to improve the patient care process. Currently, there have been considerable efforts in digital CPGs for their application to build Clinical Decision Support Systems (CDSSs) in order to deploy them in several hospitals.*

*The representation of guidelines and their introduction in Clinical Information System (CIS) can lead to efficient Clinical Decision Support Systems (CDSS), however this poses several interesting challenges as it involves problems of knowledge representation, inference, workflow definition, access to unstructured databases of medical records and others.*

*In this paper we describe the architecture of the Guideline Engine, as part of the KON<sup>3</sup> (Knowledge ON ONcology through ONtology) project. We use a semantic web approach – employing a domain ontology, a patient ontology, decision rules and a Guideline Engine formed by a Process Engine and by a Rule Engine. A Guideline Engine is a computer program which can interpret a clinical guideline represented in a computerized format and perform actions towards the user of an electronic health record (EHR). We also report a specific case study of the application of the model in oncology.*

**Index Terms** - CDSS (Clinical Decision Support System), Ontology, CPGs (Clinical Practice Guidelines), Protégé, KON<sup>3</sup> (Knowledge ON ONcology through ONtology), SAGE, Process Engine, Rule Engine, EBM (Evidence Based Medicine)

## 1. Introduction

Clinical Practice Guidelines (CPGs) are disease-specific recommendations to assist clinical decision-making in accordance with best evidence.

Several studies have demonstrated that health professionals show more effective compliance with guidelines when they are embedded in the knowledge layer of the CDSS and provide a customized management protocol for the individual patient at the point of care [13]. However, the sustainable and successful application of CPG requires a sequence of activities, in particular the CPG needs to be formally computerized to enable it to be executed by computer systems,

and to be systematically incorporated within a CDSS.

CPG guided DSS [7][14][17] are particularly useful in clinical settings where health different clinicians are required to deal with complex or unusual cases with the aim of standardize and share the knowledge about clinical treatments. In such situations, CPG-based DSS can guide the clinician's actions and suggest proper recommendations.

In order to achieve a CPG-guided CDSS is necessary to (a) encode CPGs in a structured format at semantic level (ontology); (b) transform the CPGs inherent decision logic into medically salient decision rules; (c) execute the computerized CPG to achieve decision support; (d) ensure the validity of the transformed knowledge and to provide trust in the recommended actions.

In literature, there are several theoretical approaches in CPGs systems, see for example [1] and [21]. On the contrary there are very few papers describing experiences about the realization of a CDSS that reports approaches, architectures and implementation details including issues, limitations and assumptions.

SAGE Project (Standard-based Sharable Active Guideline Environment) [24][25], involves a Guideline Model, a Protégé based tool for encoding, viewing and testing CPGs, the knowledge deployment process and the knowledge execution architecture. However no detail information about the architecture and functionalities of its Execution Engine is available in literature, except [20].

Another interesting work is reported in [12] suggesting an approach and architecture for implementing a CDSS that adopts SAGE Guideline Model. The followed approach involves the translation of the guideline representations to a proprietary knowledge model and to store them in knowledge repository.

In order to be integrated into the clinical workflow, the guideline based approach should depend on the specific patient and pathology at the hand. This implies that CDSS should be integral part of the CIS and the inference engine must be linked with all available clinical records of the patient [6]. Indeed, the wide-spread distribution and use of computable CPG content can be improved if the research community focuses on lack of standards for representing medical knowledge, and on the prohibitive complexity and expense required to adapt encoded guideline content across the heterogeneity of data structures, semantics, and medical vocabularies in use in the nation's health care information systems.

The main objective of the KON<sup>3</sup> [4], a joint effort between

companies, university and regional government agencies, is to obtain a sharable knowledge based on CPG at a reasonable cost and in a form that can be integrated into the clinician's workflow. The main features of KON<sup>3</sup> is the adoption an Ontology for representation of guidelines in addition to a registry based healthcare information infrastructure.

Here we describe the main architecture and functionalities of the central element of developed system consisting into a Guideline Engine together with its application within an oncology environment.

KON<sup>3</sup> CDSS is a generic system, because it's guideline independent provided that it's conforms to adopted Guideline Model. If this requirement is met, so it's possible to encode end execute any guideline.

The paper is organized as follows, *Section II* describes KON<sup>3</sup> project and the adopted architecture. *Section III* describes the KON<sup>3</sup> underlying knowledge representation while in *Section IV*, the architecture of the Guideline Engine is described in details. *Section V* describes a case of study in oncology environment, in particular in the Breast Cancer environment. This guideline, designed and represented at ontology level, is executed by the KON<sup>3</sup> Guideline Engine. *Section VI* contains conclusions and future works.

## 2. KON<sup>3</sup>

KON<sup>3</sup> architecture is described in this section. As shown in Figure 1, the KON<sup>3</sup> CDSS is composed by:

- **Knowledge Base**, it's the knowledge representation at semantic level. It's divided into:
  - *Guideline*, it represents the guideline model;
  - *VMR (Virtual Medical Record)*, it represents the patient data;
  - *Vocabulary*, it contains information about the used vocabulary;
  - *Expression*, it is the module to represent the rules.
- **Guideline Engine**, it is the module to guideline executions. It's divided into:
  - *Process Engine*, it's the scheduler of the guideline;
  - *Rule Engine*, it is the component that executes the rules.
- **Guideline Editor**, it provides the necessary tools to design a guideline.

The CDSS interacts with Electronic Health Record (EHR) Module through some interfaces in order to get and set data in patient's EHRs.

The Electronic Patient Record (EPR) is the record of the *periodic* care provided mainly by one institution. Typically this will relate to the healthcare provided by an hospital to a patient.

The EHR is a *longitudinal* electronic record of patient health information generated by one or more encounters in any care delivery setting. Included in this information are patient demographics, progress notes, problems, medications, vital signs, past medical history, immunizations, laboratory data and radiology reports.

The CDSS, EHR and EPR modules share a vocabulary in

order to use a single terminology. This is a recurring problem for the semantic interoperability. The main used vocabularies are SNOMED [26] and ICD9 [9].

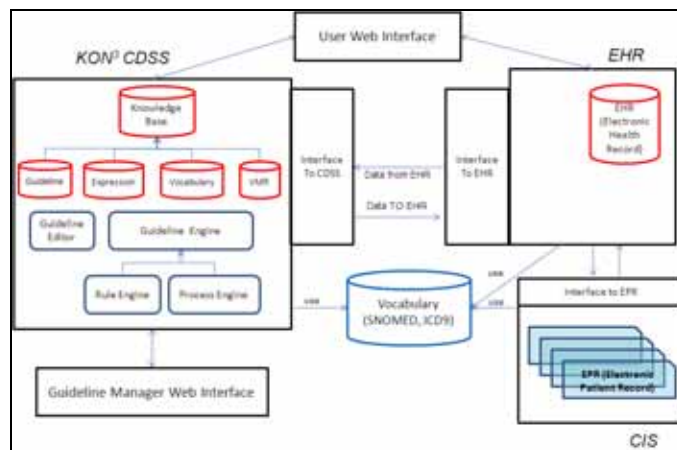


Figure 1: Kon<sup>3</sup> architecture

### 2.1. Overall picture

An overall picture describing the context on which KON<sup>3</sup> works is shown in Figure 2. KON<sup>3</sup> Guideline Engine is depicted with its knowledge base and its main modules. It is also shown a Client Module that permits clinician to interact with EHR and with KON<sup>3</sup> too.

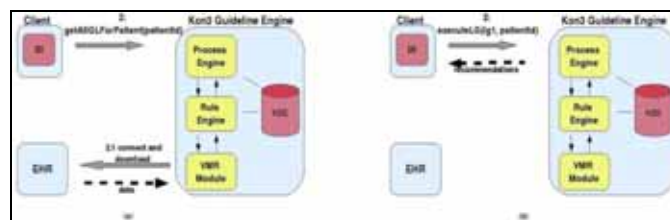


Figure 2: an overall picture

In a typical scenario, clinician inspects the electronic health record of a patient through an user interface. He also interacts with KON<sup>3</sup> CDSS, in order to extract guidelines that should be activated for the patient profile specified (Figure 2 (a)). During this phase, KON<sup>3</sup> interacts, through the VMR Module, with EHR, in order to extract needed clinical data and populate an internal VMR. After clinician and patient agree on a specific health care drift, and execute a specific guideline. So KON<sup>3</sup> CDSS influences health choices by clinicians in order to improve health care, offering some disease-specific recommendations (Figure 2 (b)).

## 3. KON<sup>3</sup> Knowledge Base

### 3.1. Guideline Model

The SAGE Guideline Model is briefly described here for illustration purposes. For a more detailed survey, see [23][25].

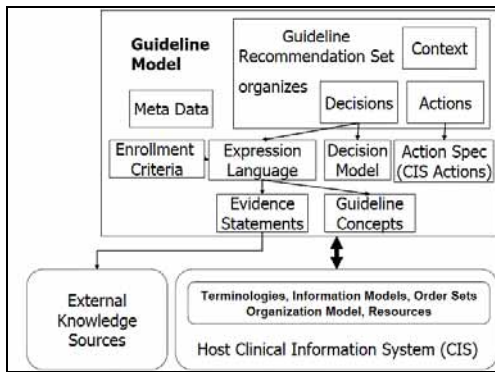


Figure 3: The SAGE Guideline Model

SAGE represents the evolution and aggregation of concepts, solutions and approaches of previous works on guideline modeling (including Asbru [15], GLIF3 [3][16], EON [23], PROforma [5], GUIDE [19] and PRODIGY [11]). It advances the state of the art by focusing on requirements that previous models have not met simultaneously: (a) incorporation of workflow awareness, (b) employment of information and terminology standards, (c) incorporation of simple flow-of-control standards, and (d) attention to integration with vendor CIS. Here we don't analyze in details the various works; for a detailed survey see [2].

The Guideline Model is a computable knowledge representation "format" for encoding the content and logic of executable CPGs.

It conceptualizes CPGs as having *metadata* such as issuing organization, *enrollment criteria* that defines its target population and *recommendation sets* consisting of some usage *context* (specific clinical circumstances) where some course of *actions* are preferred over others (*decisions* about appropriate health care).

The enrollment and decision criteria are written in terms of an executable *expression language* that make use of guideline concepts (linked to standard terminologies) and clinical evidence (formulated as *evidence statements*) for selecting particular action and that may make *queries to external knowledge sources*. Guideline actions are defined in terms of a set of *action specifications* (e.g., order laboratory tests or send an alert message) that are linked to corresponding actions in a CIS.

### 3.2. Standard Terminology

In order to support semantic interoperability at the domain level, vocabulary standards are needed. KON<sup>3</sup> uses the core vocabulary resources of SNOMED CT, in order to support semantic interoperability between CDSS and EHR.

### 3.3. Virtual Medical Record – VMR

A major obstacle in deploying and executing CPG is the variability of electronic medical records and the consequent need of adapters. It is also needed that a CDSS execution engine can query and update patient data during guideline execution regardless of data base organization. In order to provide standardization of content at the information model level, an idealized model of clinical record information

artifacts compliant with formalisms of the HL7 Reference Information Model (RIM) [8] is adopted. This is called the Virtual medical Record (VMR) [10]. The adopted VMR in KON<sup>3</sup> is composed by 13 classes: *Observation, Encounter, Problem, Adverse Reaction, VMR Order, Agent, Referral, Appointment, Alert, Composite Clinical Model, Procedure, Goal*.

### 3.4. Expression Model

SAGE Guideline Model adopts GELLO [22] as expression language; it was developed by the Clinical Decision Support Technical Committee (CDSTC) of HL7. GELLO is a generic expression language that can be used with any object-oriented data model, but is a complex string-based language that is not easy to write for someone who is not trained technically.

To make easy for guideline encoders to author computable expressions, the adopted model introduces a number of classes that organize expressions into typed data values, variables, functions, and criteria. Each expression class corresponds to a template of stereotypical GELLO expressions.

Expression Model supports four major types of criteria templates: boolean combination, comparison, existence and goal-satisfaction. In details criteria classes supported are:

- *Comparison Criterion*, a criterion used to compare an instance or a set of instances of VMR classes - retrieved according to a coded concept and a valid window - with a predefined value;
- *N-ary Criterion*, a Boolean combination of other criteria;
- *Presence Criterion*, a criterion that checks for presence or absence of coded concept in instances of a VMR class within the valid window. There are five flavors of *Presence Criterion*, one for each specific type of instances (e.g. *Observation*), defining different behaviors: *Allergy Presence Criterion, Intervention Presence Criterion, Substance Administration Presence Criterion, Observation Presence Criterion*.
- *Goal Criterion*, a criterion that checks if a goal of a measurable clinical data is satisfied.

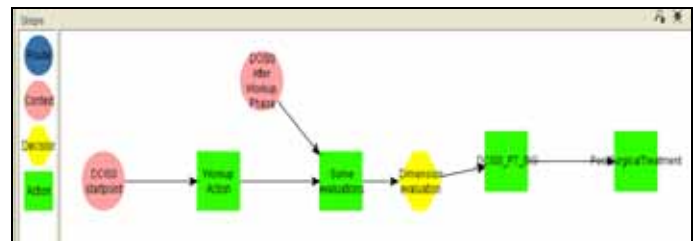


Figure 4: A guideline example design through a graph

## 4. KON<sup>3</sup> Guideline Engine

### 4.1. The Process Engine

The Process Engine is a module whose purpose is to execute, step by step, a guideline working as an action scheduler.

A guideline may be designed through a graph. An example of a guideline is shown in Figure 4; there are a set of useful elements to design a guideline, such as:

- **Context**, it defines the guideline context. It's possible, through the context, to define constraints, events and pre-conditions necessary to continue a guideline. It's also possible to entry in a subguideline;
- **Action**, it defines the current step to execute. The current step may be a set of actions (*Action Specifications*). Also the action may be formed by constraints and pre-conditions;
- **Decision**, it consists of an alternative sets. Each alternative is formed by rule sets. The Rule Engine evaluates these rules and it returns a Boolean value. The rules may be (A) *Strict Rule Out*, if the Rule Engine returns a *true* value, then it means this alternative isn't feasible, else it's; (B) *Strict Rule In*, if the rule sets with *true* value exceed a fixed threshold, then the alternative is strongly recommended; (C) *Rule Out*, if the Rule Engine return a *true* value, then it means that there's a contraindication, however the clinician can select the alternative; (D) *Rule In*, if the Rule Engine returns a *true* value, then the alternative is recommended. However, if there's a *false* value, the clinician can select the alternative.

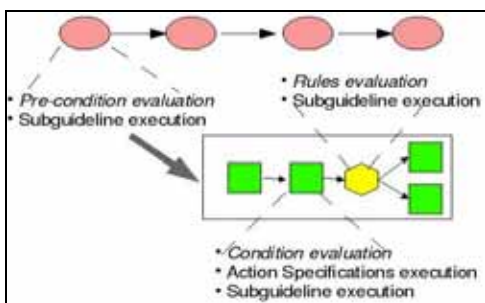


Figure 5: A typical guideline structure

In general, an expected guideline is composed by a sequence of Contexts. We adopt this convention in order to manage correctly execution state to allow at clinician to entry, in any point of the guideline. This necessity is derived by real world, e.g. a patient should start a care in a hospital and should continue it in another. So, we model these other entry points as *Contexts*, too.

Process Engine traverses a guideline graph and manages its single step. Process Engine executes each type of steps with different behavioral, see Figure 5.

If it's a *Context*, then the Process Engine controls if there are conditions to evaluate. In this case it calls the Rule Engine for evaluation of them. If the Rule Engine returns a *true* value, then this step is activated, and so the Process Engine entries in a possible subguideline. Also a subguideline is composed by other actions and decisions.

If it's an *Action*, then the Process Engine controls if there are conditions to evaluate and, if required, calls Rule Engine. If the Rule Engine returns a *true* value, then this step is activated, and so the Process Engine manages the *Action Specifications*. *Action Specifications* are action sets, in particular they could be classified in: (a) *Notify Action*, which purpose is to send a textual message to the clinician; (b) *Display Action*, which purpose is to display expressions,

clinical data sets and supplemental materials; (c) *Inquire Action*, which purpose is to interact with the clinician to get some information about the patient; (d) *Recommended VMR Order*, which purpose is to set interventions, referrals and exams into the CIS.

If it's a *Decision*, then the Process Engine controls if there are conditions to evaluate and, if required, calls the Rule Engine. The Process Engine evaluates first the *Strict Rule Out*. If at least one is *true*, then the alternative is discarded, else the other rules are evaluated. It also determines recommended alternatives, according to *Strict Rule In* threshold and highlights them. Finally all possible alternatives are returned and proposed to clinician.

## 4.2. The Rule Engine

Since today there isn't a mature, complete and open source GELLO expression engine accessible via programmable interface, we choose to design and develop an *ad-hoc* engine.

The Rule Engine developed works on the expression classes defined in the Expression Model, in particular on the evaluation of the expression templates, called criteria. It is possible to define a criterion as an access precondition on *Context*, *Action* and *Action Specification* nodes and also as a decision rule on *Decision* nodes, such as *Strict Rule In*, *Strict Rule Out*, *Rule In*, *Rule Out*. Rule Engine supports and executes all four major types of criteria templates. For each class supported, there is an handler module that manages and resolves instances of it. As mentioned above, an interaction between Rule Engine and VMR module (and from VMR module and EHR, too) is strongly needed, in order to retrieve VMR instances needed to evaluate a criterion.

In a typical scenario, Guideline Engine asks to Rule Engine to resolve a particular criterion on a specified patient, e.g. a condition for evaluating if a particular *Context* may be activated. Rule Engine retrieves needed clinical data for specified patient from the VMR and resolves it. The output could be *true* (condition satisfied for the patient) or *false* (condition unsatisfied for the patient).

## 4.3. VMR Module

This module manages patients data collecting them into the VMRs. When asked by Rule Engine, it retrieves data for a specific patient from EHR, then it extracts only needed data and populates VMR. Finally, it offers the VMRs to Rule Engine.

## 5. A Case of Study: DCIS (Ductal Carcinoma In Situ)

The case of study described here, is extracted and modeled from a real guideline from NCCN<sup>1</sup> Clinical Practice Guidelines in Oncology v.2.2007. The specific guideline is the Ductal Carcinoma In Situ (DCIS) one.

The National Comprehensive Cancer Network (NCCN), a not-for-profit alliance of 21 of the world's leading cancer centers, is dedicated to improving the quality and effectiveness of care provided to patients with cancer. NCCN Member

<sup>1</sup> NCCN web site: [www.nccn.org](http://www.nccn.org)



Institutions develop resources, such as guidelines, that present valuable information to the numerous stakeholders in the health care delivery system.

DCIS refers to the most common type of noninvasive breast cancer in women. *In situ*, or "in place", describes a cancer that has not moved out of the area of the body where it originally developed. With DCIS, the cancer cells are confined to milk ducts in the breast and have not spread into the fatty breast tissue or to any other part of the body (such as the lymph nodes).

The clinical scenarios and guideline logic are encoded into a computer interpretable model of guidelines, using a Protégé [18] plug-in, called Graph Widget.



Figure 6: The DCIS guideline encoded.

The DCIS guideline, as shown in Figure 6, is composed by four main phases:

- WorkUp (“DCIS startpoint”);
- Primary Treatment (“After Workup”);
- PostSurgical Treatment;
- Follow Up.

### WorkUp

The guideline startpoint, the first possible *Context*, is “DCIS startpoint”. The specific precondition is “Female and DCIS”, it is composed by two criteria: (1) A “DCIS diagnosed” for this patient, (2) and “Patient is Female”. So the precondition is *true* only if either are *true*. If this *Context* is enabled the execution could start with the subguideline. In “DCIS startpoint” subguideline there is only an *Action*, called “Workup Actions”, that involves notifies that an historical and physical examination, titled “H&P to do”, and a “Pathology Review” are necessary. Also two recommended clinical order titled “Mammography” and “Determination of Tumor Estrogen” should be performed.

### Primary Treatment

The second phase is Primary Treatment (PT), it is composed by the subguideline shown in Figure 7. The precondition for access here is composed by the precondition for “Workup Action” *Context* and the criterion “Workup Action Completed”.

Last one criterion is in order to analyze the VMR and verify that the needed clinical exams are performed in a predefined valid window.

In PT subguideline there is an inquire step (“Some evaluations” *Action*), for asking to clinician about DCIS details, such as DCIS grade and DCIS unicentricism. After a *Decision* step is performed, in order to select the recommended Primary Treatment for the patient. Here clinician responses and other VMR data (e.g. DCIS dimension) are evaluated. A summary of the underlying rules in “Dimension Evaluation” is shown in Table I.

After this step, clinician and patient select the preferred Primary Treatment from the list (*Action* DCIS0\_PT\_SMALL or DCIS0\_PT\_BIG). Examples of possible Primary Treatment

for DCIS0\_PT\_BIG are “Mastectomy without lymph node dissection with/without reconstruction” and “Lumpectomy with RT”.

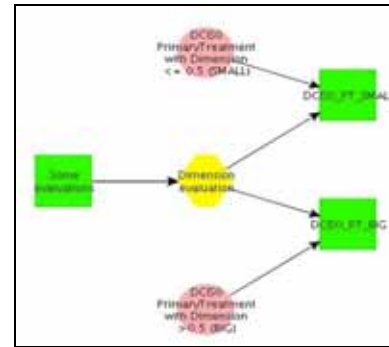


Figure 7: Primary Treatment in DCIS

	Strict Rule Out	Strict Rule In	Threshold
DCIS0_PT_SMALL	DCIS DIMENSION >= 0.5 cm	Is unicentric? Is low grade?	1
DCIS0_PT_BIG	DCIS DIMENSION < 0.5 cm	-	-

Table 1: Alternatives in Primary Treatment

### PostSurgical Treatment

This phase involves the post surgical treatment, see Figure 8. In particular is determined if a specific PostSurgical Treatment is needed (“DCIS PostSurgical Treatment” *Action*), or if only the risk reduction therapy is needed, see Table II. This is determinate evaluating previous chooses, inferred from VMR (e.g. “Mastectomy” preferred as choice in Primary Treatment phase and detected as performed in VMR). Either actions are composed by some *Actions Specifications* that involves notifies to clinician, displays data, displays supplemental materials (such as other guidelines, link to publications, evidences, statistics), etc.

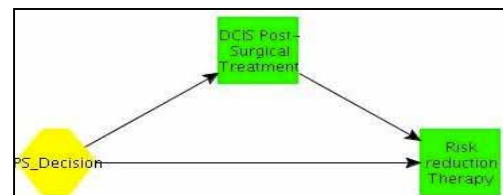


Figure 8: Subguideline in PostSurgical Treatment Context.

	Strict Rule Out	Strict Rule In	Threshold
DCIS0 PostSurgical Treatment	-	PT: Lumpectomy and RT PT: Mastectomy	1
DCIS0 Reduction Therapy	-	-	-

Table 2: Alternatives in Postsurgical Treatment

### Follow Up

The prerequisite to access into this *Context* is a criterion composed by the PostSurgical Treatment precondition and the termination condition of the previous phase. This termination

condition is inferred from the VMR in a manner quite similar to previous ones.

In the FollowUp subguideline the required follow up exams are shown to clinicians specifying administration time interval, amount and frequency. The exams are determinate based on previous chooses, such as choices made in Primary Treatment and PostSurgical Treatment (e.g. if treated with Tamoxifen during PostSurgical Treatment, then display also, as supplemental material, “NCCN Breast Cancer Risk Reduction Guideline”).

## 6. Conclusions and Future Works

In this paper KON<sup>3</sup> CDSS is described, focusing on the execution of a CPGs in oncology environment (in particular in Breast Cancer environment). However, the described system could also be applied to other domains where the activities are based on a design that entails a decision logic that is structured in an algorithmic format. In general the algorithmic format should be formed by Action Step, Decisional Step, and Context Step.

We described a Guideline Engine composed by a Process Engine and a Rule Engine to satisfy above requirements. The Process Engine is the actions scheduler, while the Rule Engine evaluates the criteria contained in a guideline. These criteria are used in order to define conditions and decision rules.

Future developments will involve: (a) *CPG Process Authoring*. It will provides a Graphical User Interface (GUI) that permits to model and formalize CPGs; (b) *CPG Rule Authoring*. It will provides a GUI that permits to define rules in a CPG; (c) *Use of EBM (Evidence Based Medicine)*. KON<sup>3</sup> purpose is to provide a complex system to support clinicians in their decision during a process care. The support is not only through a guideline execution, but also through EBM theory. Through EBM, the clinician is supported through a set of documents, experiences, statistics.

## 7. Availability

All the project documents, as Guideline Model documentation, KON<sup>3</sup> architecture documentation and the prototype are available on our web site <http://www.koncube.org>.

## References

- [1] Beal T, Heard S., “An Ontology-based Model of Clinical Information”, MEDINFO2007, IOS Press.
- [2] Boxwala A., Greenes R.A., Shortliffe E.H. et al., 2000, “Toward Standardisation of Electronic Guideline Representation”. *MD Computing*, vol. 17(6), p. 39-44.
- [3] Boxwala AA, Peleg M, Tu SW, Ogunyemi O, Zeng Q, Wang D, et al., 2004, “GLIF3: A Representation Format for Sharable Computer-Interpretable Clinical Practice Guidelines”. *J Biomed Inform.* Vol. 37(3), p. 147-161.
- [4] Ceccarelli M, Donatiello A, Vitale D, 2008, “KON3: a Clinical Decision Support System, in oncology environment, based on knowledge management”, 20th IEEE ICTAI 08, Vol. 2, p. 206-210.
- [5] Fox J, Das SK., 2000, *Safe and Sound*, Cambridge: MIT Press.
- [6] Goldstein M.K. et al., 2004, “Translating Research into Practice: Organizational Issues in Implementing Automated Decision Support for Hypertension in Three Medical Centers”, *Journal of Am. Med. Informatics Assoc.*, vol. 11, p. 368-376.
- [7] Gray J. A. Muir, 1997, *Evidence Based Healthcare*, W.B. Saunders Company.
- [8] HL7 Reference Information Model (RIM), [www.hl7.org/Library/data-model/RIM/modelpage\\_mem.htm](http://www.hl7.org/Library/data-model/RIM/modelpage_mem.htm)
- [9] ICD9, <http://www.who.int/classifications/icd/en/>
- [10] Johnson PD, Tu SW, Musen MA, Purves I, 2001, “A Virtual Medical Record for Guideline-Based Decision Support”, Proc. AMIA Symp., p. 294-298.
- [11] Johnson PD, Tu SW, Booth N, Sugden B, Purves IN, 2000, “Using Scenarios in Chronic Disease Management Guidelines for Primary Care”. In *Overhage*, JM editor. Proc AMIA Symp, p. 389-393.
- [12] Kim J Ah, Cho I, Kim J., 2008, “CDSS (Clinical Decision Support System) Architecture in Korea”. In Proc. International Conference On Convergence and Hybrid Information Technology; p. 700-703.
- [13] Lobach, D.F., Hammond, W.E., 1997, Computerized Decision Support Based on a Clinical Practice Guideline Improves Compliance with Care Standards. *Am J Med.*
- [14] McGlynn E.A., Asch S.M., Adams J., Keesey J., Hicks J., DeCristofar A., and Kerr E.A., 2003, “The quality of health care delivered to adults in the United States”, *New England Journal of Medicine*. Vol. 348(26), p. 2635-2645.
- [15] Miksch S, Shahar Y, Johnson P., 1997, “Asbru: A Task-Specific, Intention-Based, and Time-Oriented Language for Representing Skeletal Plans”. In: Motta E, Harmelen, F. v., Pierret-Golbreich, C., Filby, I., Wijngaards, N., editor., KEML-97, p. 9-1-9-20.
- [16] Peleg M, Boxwala AA, Tu SW, Zeng Q, Ogunyemi O, Wang D, et al., 2004, “The InterMed Approach to Sharable Computer-Interpretable Guidelines: A Review”. *J Am Med Inform Assoc.*, Vol. 11(1) p. 1-10.
- [17] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R.A. Greenes, R. Hall, P.D. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E.H. Shortliffe, and M. Stefanelli, 2003, “Comparing computer-interpretable guideline models: a case-study approach,” *Journal of the American Medical Informatics Association*, Vol. 10(1), p. 52-68
- [18] Protégé Official Web Site, <http://protege.stanford.edu/>
- [19] Quaglini S, Stefanelli M, Cavallini A, Micieli G, Fassino C, Mossa C., 2000, “Guideline-Based Careflow Systems”. *Artif. Intell. Med.*, Vol. 5(22) p. 5-22.
- [20] Ram Prabhu; Berg David; Tu Samson; Mansfield Guy; Ye Qin; Abarbanel Robert; Beard Nick, 2004, “Executing clinical practice guidelines using the SAGE execution engine”. *Studies in health technology and informatics*, 107(Pt 1) p. 251-5.
- [21] Siddiqi J. et al., 2006, “Towards and Automated Diagnosis for the Treatment of Colon Cancer: Position and Progress”, IEEE AICS2006, IEEE Press. <http://www.match-project.com/>
- [22] Sordo M, Ogunyemi O, Boxwala AA, Greenes RA, Tu S., 2004, “GELLO: An Object-Oriented Query and Expression Language for Clinical Decision Support”. Summary Report prepared for OpenClinical.
- [23] Tu SW, Musen MA., 1999, “A Flexible Approach to Guideline Modeling”. Proc AMIA Symp 1999; *Hanley & Belfus, Inc.*, p. 420-4.
- [24] Tu SW, Campbell JR, Glasgow J et al., 2007, “The SAGE Guideline Model: Achievements and Overview”. *J Am Med Inform Assoc.* 2007 Vol. 14(5), p. 589-98.
- [25] Tu SW, Glasgow J, 2006, “SAGE Guideline Model Technical Specification”, SAGE Consortium.
- [26] Wang AY, Sable JH, Spackman KA, 2002, “The SNOMED clinical terms development process: refinement and analysis of content”, Proc AMIA Symp 2002, p. 845-849.

# Ontology-Based Semantic Annotations of medical articles

Jihen Majdoubi — Mohamed Tmar — Faiez Gargouri

MIRACL Laboratory, ISIM Institute, BP 1030-3018, Sfax, Tunisia

E-mail { majdoubi\_jihen@yahoo.fr, mohamed.tmar@isimsf.rnu.tn, faiez.gargouri@fsegs.rnu.tn }

## Abstract

This paper proposes a semi-automatic method using domain ontology for generating the semantic annotation for medical articles. First, our approach uses NLP (Natural Language Processing) techniques to extract the candidate terms. Second, these terms are weighted based on their frequencies, their locations in the article and their semantics. Next, the user evaluates the frequent terms that do not exist in the ontology to choose the terms will be exist in the annotation result. Finally, the structured result annotation is build.

## 1. Introduction

In the medical field, scientific articles represent a very important source of knowledge for researchers of this domain. These researchers usually need to deal with a large amount of scientific and technical articles for checking, validating and enriching their research work. But due to the large amount of scientific article published on the web, an efficient detection and use of this knowledge is quite a difficult task. The semantic web can facilitate this task: it can be carried out by associating to each scientific article a semantic annotation. This annotation provides a more precise description of the knowledge contained in the document in order to facilitate exploitation and access of this document.

During the few last years, the research community has devoted an increasing effort in the semantic annotation of medical articles [1][5][7]. However, with these approaches, the annotation's concepts must belong to the Terminological-Ontological-Resources (TOR). Consequently, a concept (relevant to the field) extract of the document but nonexistent in the TOR, does not appear in the annotation result.

We can remark the gap between the goal of semantic annotation and the method that achieves it: the major goal of semantic annotation is to determine the specific document's content but the semantic annotation extracts the ontology's concepts appearing in the document. This lack of methods motivated us to explore a new track leading to an annotation that represents the content of the document rather than the ontology's concepts contained in the document.

Our proposed methodology is mainly based on the weighting term; non-existent term in the ontology but having a weight that exceeds the fixed threshold can exist in the annotation result if the expert judges that this term

is representative of the document. Moreover, the term can be added to our ontology if the expert recommends that the updating of ontology is necessary.

Moreover, the majority of research works in semantic annotation of medical articles are generally based on the content of the document. In deed, few works have integrated the structure of the document in the process of indexation. Like [11] and [13] which assigned an additional weight to the terms that are extracted from the title or the subtitle of the document. However, these works doesn't take into account the semantic. This lack motivated us to explore an approach combining the content, the structure and the semantic of document to be annotated.

The remainder of this paper is organized as follows. Section 2 overviews our approach context. Section 3 presents the step of extraction term. Section 4 illustrates the phase of weighting terms. Section 5 indicates how the annotation is built. Finally, section 6 summarizes our proposal and outlines future work related to our annotation methodology as a whole.

## 2. Our approach

Our objective is to develop an approach which, starting from a medical article and domain ontology, generates a semantic annotation that describes the article's content.

Figure 1 describes the general architecture of our annotation's method.

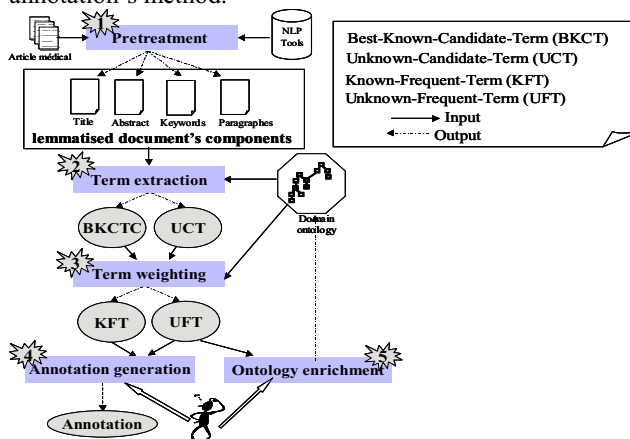


Figure1. General architecture of our annotation's method

We present the architecture's components in the following subsections.

## 2.1. Domain ontology

In our approach, we denote Ontology by (O) characterized by a set of concepts (C) and a set of relationships (R) between these concepts. Each concept ( $c \in C$ ) is represented by a term called “label” and a set of its synonyms. Each term is represented by a set of attributes where each attribute is a single term.

$C = \{c_1, c_2, \dots, c_n\}$ ,  $c_i = (\text{label}_i, \{s_{i1}, s_{i2}, \dots, s_{in}\})$ ,  $s_{ij} = (\text{att}_{ij1}, \text{att}_{ij2}, \dots, \text{att}_{ijk})$ . For example,  $c_1 = (\text{infection of ear}, \{\text{otit}, \text{infection of ear}, \text{infection bacterial of ear}, \text{inflammation of ear}, \text{inflammation bacterial of ear}\})$ . Thus,  $c_1 = ((\text{infection of ear}), \{(\text{otit}), (\text{infection of ear}), (\text{infection bacterial of ear}), (\text{inflammation of ear}), (\text{inflammation bacterial of ear})\})$ .

Formally, the relationships (R) is a set of triplets  $(c_i, c_j, r)$  where  $c_i$  and  $c_j$  are two concepts related by  $r \in \{\text{hypernymy}, \text{hyponymy}, \text{meronymy}, \text{holonymy}\}$ .

$R \subset C \times C \times \{\text{hypernymy}, \text{hyponymy}, \text{meronymy}, \text{holonymy}\}$

In the following, we detail these relationships:

- *hypernymy*: the generic concept used for the is-a relation (cancer is *hypernym* of Cancer of blood).

- *hyponymy*: the specific concept used for the is-a relation (Cancer of blood is *hyponym* of cancer).

*Y is hypernym of X if X is hyponym of Y.*

- *holonymy*: the global concept used for the *has-a* relation (face *has-a* {eyes, nose}, face is *Holonym* of eyes).

- *meronymy*: the concept which is part of another concept (nose is *meronym* of face).

*Y is holonym of X if X is meronym of Y.*

## 2.2. Pretreatment (Fig 1-(1))

As mentioned previously, the the document’s structure will be used to weigh the document’s terms. Thus, in this step our system extracts the various document components (title, abstract, keywords and paragraphs). After, for each one of these components, we prune the stop words. Then we use the Tokeniser module of GATE [2] in order to split the document’s components into tokens. Finally the TreeTagger stems these tokens to assign a grammatical category (noun, verb...) and lemma to each token.

## 2.3. Term extraction (Fig 1-(2))

In the majority of annotations approach, the extraction term is only based on the TOR, what we consider insufficient because this TOR (ontology, thesaurus) may be restricted with the field. Therefore, a concept (relevant to the field) extract of the document but non-existent in the TOR, does not appear in the annotation result.

In contrast to these approaches, in our approach we index the terms independently of any reference ontology.

Two alternative ways can be distinguished. The first one consists in Best-Known-Candidate-Term (BKCT) which forms the ontology’s concepts existing in the text to be indexed. The second way represents the Unknown-Candidate-Term (UCT) which contains the terms that inexist in ontology but can be interesting for the domain.

## 2.4. Weighting terms (Fig 1-(3))

The extracted terms (BKCT, UCT) are then weighted, to determine the frequent terms. These frequent terms are divided between the (1) known-Frequent-Terms (KFT) which contains the frequent ontology’s terms and (2) Unknown-Frequent-Terms (UFT) that represents the frequent terms that do not exist in the ontology.

## 2.5. Annotation generation (Fig 1-(4))

This step consists in collecting any information coming from previous phases in order to generate an annotation describing the content of the document.

## 2.6. Ontology enrichment (Fig 1-(5))

In order to maintain ontology’s coherence and to enrich the knowledge represented in the ontology, we exploit the information extracted from scientific articles and more precisely, we use the set of UFT.

In the remainder of this paper, we focus on the three steps: *Term extraction, weighting terms and Annotation generation*. For further details about the *Pretreatment module*, the reader is referred to [8] and [9].

## 3. Term extraction

Term extraction is a two-phase process: (1), extract the BKCT and (2) determine the UCT.

In our approach, to extract the ontology’s concepts we use the vector space model [12] which measures the documents similarities with the query.

We have adapted this model by substituting the document by the sentence and the query by the terms of ontology’s concept.

Each stemmed sentence S is a list of stems that are ordered in S as they appear in the text.  $S = (t_1, t_2, \dots, t_n)$ .

Each concept  $c_i$  is processing with TreeTagger in order to return for each one of its term  $s_{ij}$  ( $s_{ij} = (\text{att}_{ij1}, \text{att}_{ij2}, \dots, \text{att}_{ijk})$ ), its lemmatized form.

Thus,  $\text{sim}_s(S, s) = \max_{\substack{i \in \{1, 2, \dots, n\}, j \in \{1, 2, \dots, m\} \\ k \in \{1, 2, \dots, \inf(n-i, m-j)\} \\ (t_i, t_{i+1}, \dots, t_{i+k}) = (\text{att}_{j1}, \text{att}_{j+1}, \dots, \text{att}_{j+k})}} \dots$

For a concept c composed of n terms, its similarity  $\text{Sim}_s(S, c)$  in a sentence S is defined as flows:

$$\text{Sim}(S, c_i = (\text{name}_{c_i}, \{s_{i1}, s_{i2}, \dots, s_{in}\})) = \sum_{s \in \{s_{i1}, s_{i2}, \dots, s_{in}\}} \text{Sim}_s(S, s)$$

The process of term extraction is iterative. For each stemmed sentence, we compute its score with the set of ontology’s concepts. Only those having a strictly positive score are proposed to the system like Known-candidate-term (KCT).

$$\exists c \in C \wedge \text{Sim}(S, c) > 0 \Rightarrow c \in \text{KCT}(S).$$

However, a sentence can contain several KCT, Which is or are the Best Candidate-Term-known (BKCT(S)). Two cases are dealt with in this situation.

Let us consider: a sentence S and  $|\text{KCT}(S)| = k$  with  $k > 1$ . Which is or are the BKCT(S) among these concepts?

**Case1: disjoint concept.**

This case occurs when the concept is disjoint with the other KCT(S). Thus, this concept will be qualified like BKCT(S).

In order to determine for each concept c its type (disjoint or joint), we use the function *Listcommon* (*Listcommon: Sentence, concept → concept*). *Listcommon* associates for each concept c of KCT(S), all concepts (c1,..., ci) of KCT(S) that have one or more common terms with c.

$$\exists c \in KCT(S) \wedge Listcommon(S,c) = \emptyset \Rightarrow isdisjoint(c,S).$$

Thus, the case1 results in the following assertion

$$\exists c \in KCT(S) \wedge isdisjoint(c,S) \Rightarrow c \in BKCT(S)$$

**Case2: joint concept.**

This case occurs when a concept has one or more common terms with at least the one other KCT(S). The concept which has the highest score with the sentence S will be qualified like BKCT(S).

$$\exists c \in KCT(S) \wedge |Listcommon(S,c)| > 0 \Rightarrow isjoint(c,S).$$

In order to determine the concept which has the highest score in the sentence S, we use the function *MaxScor* (*MaxScor: concept → concept*). *MaxScor* associates for a set of concepts (c1, c2,...,cn) , the concept that has the highest score.

The case 2 results in the following assertion

$$\exists c \in KCT(S) \wedge isjoint(c,S) \wedge ci = MaxScor(Listcommon(S,c)) \Rightarrow c \in BKCT(S)$$

Thus, each term not recognized like the term’s attribute of KCT(S), will be qualified like a UCT(S).

Finally, we generate the Best-Known-Candidate-Term (BKCT(D)) and the Unknown-Candidate-Term (UCT(D)) of the document D.

$$BKCT(D) = \bigcup_{si \in S} BKCT(si)$$

$$UCT(D) = \bigcup_{si \in S} UCT(si)$$

These extracted terms (BKCT(D), UCT(D)) are then weighted in order to determine the importance of each one in the document and select the terms will be part of the annotation result.

**4. Weighting terms**

To calculate the term’s weight, the majority of approaches are based (1) either on the content of the document [4][10] in the case of textual documents, or (2) either on the combination of content and structure in the case of structured documents [14]. However, semantic is rarely taken into account.

In this paper we propose a new term weighting technique which takes into account the occurrence of the term, its location and its semantic. To do this, we use two measures: the Content-Structure-Weigh (CSW) and the Semantic-weigh (SW).

**4.1. Content-Structure-Weight**

We can notice that the frequency is not a main criterion to calculate the CSW of the term T. In deed, the CSW takes into account the term’s frequency and especially the

location of each one of its occurrences. For example, the term in “Title” will give a considerable importance (\*10) to the term that exists in “Paragraphs” (\*2). The table 1 gives the various coefficients used to weight the term’s locations. These coefficients were determined in an experimental way in [6] and also used in [3]. Thus, in the article D containing n different terms, for a given term T<sub>i</sub> (with i in [1..n] ), the CSW(T<sub>i</sub>, D) is determined as the sum of the location’s weights associated with the each occurrence of the term T<sub>i</sub> in document D.

This calculus is shown in flowing formula

$$CSW(T, D) = \sum_{i=1}^{|D|_r} f(T, D, i) \times M_i \quad \text{With:}$$

- f(T,D,i): the occurrence’s number of the term T in document D at location i with i ∈ {title, keywords, abstract, paragraphs},
- M<sub>i</sub>: weight of the position i of the term T in a document D (see Table 1),
- |D|<sub>r</sub>: the number of occurrences of term T in document D,

Table1. Coefficients of weighting

Term’s location	Weight of location
Title	10
Keywords	9
Abstract	8
Paragraphs	2

**4.2. Semantic-Weight**

In this step, we are interested in the BKCT(D) because we exploit the ontology’s relationships to calculate the Semantic-Weight. Let us recall that a BKCT(D) is composed by a set of ontology’s concept. Thus, we calculate the Semantic-Weight of concept and not the Semantic-Weight of term, but for reasons of simplicity we use the notation SW(T,D) to represent Semantic-Weight of concept. The Semantic-Weight of the concept C depends on its *hypernyms*, *hyponyms*, *meronyms* and *holonyms*.

To do this, we use the functions *List: concept, concept, relationship → concept*. *List* associates for a given concept c, a set of concepts (c1, c2,...,cn) and a relationship r, all concepts satisfying r among this set.

To calculate the SW(T,D), we apply the following assertion:

$$SW(T, D) = \frac{\sum_{g \in List(T, BKCT(D), R)} CSW(g, D) \times WR}{|List(T, BKCT(D), R)|}$$

With:

R ∈ {hyperonymie, hyponymie, meronymie, holonymie} , WR: weight that measures the importance of each one of these relationships (*hypernymy*, *hyponymy*, *meronymy* and *holonymy*). WR has a value in the range [0, 1].

For a given term T, its Term frequency (tf (T, D)) is determined as the sum of the both. Formally:

$$tf(T,D)=CSW(T,D)+SW(T,D)$$

Once the Term Frequency of term is calculated, it is used to calculate the weight of a term T in a document D (W (T, D)). Formally:  $W(T, D) = tf(T, D) \cdot \ln(N/df)$

Where N is the total number of documents and df (document frequency) is the number of documents a term T occurred in.

## 5. Annotation generation

At this point, we have on the one hand a set of Known-Frequent-Term (KFT(D)) and on the other hand a set of Unknown-Frequent-Term (UFT(D)). The annotation generation is done in two phases.

### 5.1. Analyze of UFT(D)

This step aims at to identify the really relevant terms among the set of UFT (D). For that, we find that the intervention of the user throughout this set of UFT is indispensable. In deed, the user is the only one who is able to judge whether if the term is relevant or not and to choose among the list of unknown terms, those which must appear in the annotation result.

### 5.2. Generation of the annotation result

We associate the KFT(D), with those which were chosen by the user like relevant at the previous phase, to generate a first version for the annotation result

## 6. Conclusion

In this paper we outlined a semi-automatic approach for representing the semantic content of medical articles. This approach comprises a number of advantages according to the traditional annotation methods.

- The weighting of a term depends not only on its term frequency but also on its location and its semantic.
- Annotation result contains not only the ontology's concepts but also the frequent terms that do not exist in the ontology but which are relevant according the administrator.

While the system of annotation is implemented, we are currently working on the evaluation of our method on a medical corpus in order to measure its relevance. In addition, we are implementing the ontology enrichment module

## 6. References

[1] Blott, S., Gurrin, C., Gareth, J., Jones, F., Smeaton, A., Sodrings, T. "On the Use of MeSH Headings to Improve Retrieval Effectiveness", in The Text REtrieval Conference TREC Genomics Track (TrecGen), 2003.

[2] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V. "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. ACL'02. (2002).

[3] Desmontils, E., Jaquin, C. "Indexing a Web site with a terminology oriented ontology" in The Emerging Semantic Web, IOS Press, ISBN 1-58603-255-0, pp 181-197, 2002.

[4] Erdmann, M., Aedche, A., Schnurr, H., Staab, S. "From manual to semi-automatic semantic annotation: About ontology-based text annotation tools", in Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content, P. Buitelaar, K. Hasida (Eds.).

[5] Fabrice, C., Stephen, B., Alan, F. "On Combining Text and MeSH Searches to Improve the Retrieval of MEDLINE documents" in CORIA 2006.

[6] Gamet, J. "Indexation de pages web", Report of DEA universit  de Nantes, 1998.

[7] Kayaalp, M., Aronson, A., Humphery, S., Ide, N., Tanabe, L., Smith, L., Demner, D., Lone, R., Mork, G., Bodenreider, O. "Methods for Accurate Retrieval of MEDLINE Citations. In Functional Genomics ", in NIST Special Publication 500-255 : The Twelfth Text REtrieval Conference (TREC 2003), E. M. Voorhees et L. P. Buckland ( ds), USA, NIST, p. 441-450, 2003.

[8] Majdoubi, J., Gargouri, F. "Vers une nouvelle approche d'annotation s mantique des articles scientifiques : application au domaine m dical", in SIIE'2009 " Syst mes d'Information et Intelligence Economique" du 12 au 14 f vrier 2009, Hammamet, Tunisie.

[9] Majdoubi, J., Tmar, M., Gargouri, F. "Indexing a medical article with a Terminology Oriented Ontology", In ICTA 2009 : International conference "Information and Communication Technologies and Accessibility" mai 2009, Hammamet, Tunisie.

[10] Paralic, J., Kostial, L. "Ontology-based Information Retrieval ", in Proceedings of the 14th International Conference on Information and Intelligent Systems, ISBN 953-6071-22-3, pp 23-28, 2003.

[11] Pouliquen, B. "Indexation de textes m dicaux par indexation de concepts, et ses utilisations". Th se de doctorat, Institut National des Sciences Appliqu es de Rouen, Universit  Rennes 1, 2002.

[12] Salton, G. "The SMART Retrieval System - Experiment in Automatic Document Processing. Englewood Cliffs, NJ : Prentice-Hall.

[13] Soualmia, LN., Golbreich, C., Darmoni, S. "Representing the MeSH in OWL : Towards a semi-automatic Migration", in First International Workshop on Formal Biomedical Knowledge Representation, collocated with KR 2004. p. 1-12. Whistler, Canada.

[14] Zargayouna, H., Salotti, S. "Mesure de similarit  s mantique pour l'indexation de documents semi-structur s", in 12 me Atelier de Raisonement   Partir de Cas, 2004.

# Automating Business Intelligence Recovery from a Web-based System

Jian Kang, Jianzhi Li, Jianchu Huang, Yingchun Tian and Hongji Yang  
Software Technology Research Laboratory  
School of Technology  
De Montfort University  
LE1 9BH, Leicester, England  
{jkang, jianzhili, jhuang, ytian, hyang@dmu.ac.uk}

## Abstract

*The theme of this paper is to explore a path to recover business intelligence automatically from a Web-based system to business intelligence base. It is a reverse engineering task of decomposing the program code, eliciting business intelligence data, and representing the recovered results. The process is composed of four procedures: business intelligence base decomposition; programming style-based program partition; business intelligence concept recovery; and formal business intelligence concept analysis.*

*After a brief introduction of major issues covered by the paper, the state of art of the area coined by the authors as “business intelligence elicitation from a Web-based system”, in particular, the kinds of business intelligence that can be elicited from a Web-Based system and the corresponding reverse engineering technical solutions are presented.*

**Keywords:** Business Intelligence Recovery, Web-Based System, Reverse Engineering, Program Comprehension, Program Partition, Concept Recovery, Formal Concept Analysis

## 1. Introduction

The properties of modern human life are continuously digitalised, e.g. publishing, advertising, and shopping. It indicates that computing is becoming more and more universal. With the further development of computing technology, a mobile retailer, for instance, which is concentrating on marketing mobile services and devices, might require computing modules, such as Operator and Producer News Monitoring, Service Usage Monitoring, Customer Preference Investigating, and Mobile-Plan Analysing etc., to be integrated into an existing Web-based system. This kind of computing system is named “online mobile retailing system”. The act of analysing the system, developing and integrating new modules to meet the business requirements is called software evolution.

However, the evolution task is more than analysing, developing and integrating. The overall scenario is the “online mobile retailing system” is mainly composed of three parts: retailing web site, service running system, and a database; each part serves a business role in mobile retailing; and the retailing business data is exchanging

among different parts. To ensure the software evolution task is successful, one of the key issues is that business intelligence, which is embedded in the retailing system, must be recovered and reused to meet the requirements for new module development and integration.

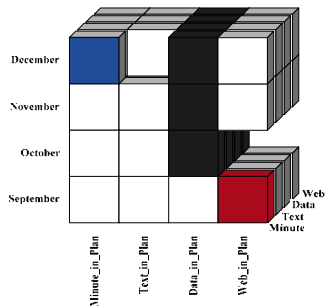
Business Intelligence (BI) [16] is not a new concept in software industry. It is a mature technology to monitor, analyse and enhance business performance. The concept of Business Intelligence Recovery (BIR) [9, 8] was introduced into software reverse engineering by Kang et al. They refers business intelligence recovery as one of the software reverse engineering activities to have business intelligence data, algorithm, mathematics model, physics model and business logic. Kang et al. argue that business intelligence recovery would help to enrich business oriented program comprehensibility for a rule-based reverse engineering system, e.g., business rule extraction system [7]. By business intelligence recovery, it would leverage reverse engineering from source code and program structure level to higher abstraction levels like component model level, software architecture level and requirement level.

This research strives to bring reviewer closer to the goal of Automating Business Intelligence Recovery (BIR) from a Web-based System. It vertically elicits business intelligence from system to Business Intelligence Base (BIB). The overall recovery process of BI is composed of four procedures: BIB decomposition, programming style-based program partition, BI concept recovery, and formal BI concept analysis. Nonetheless, the BIB glossaries must be discussed in advance. Figure 1 is an example of BIB about “one customer’s mobile service-consuming in last quarter of 2008” in the mobile retailing system.

This BIB is grouped by BI cubes, in which monthly service-consuming information of the mobile account is recorded. It is the research goal to have this BIB by BIR.

In this BIB, there are:

- 3 Dimensions: time dimension, Service dimension, and Usage Dimension, e.g., {[BIB], ([time], [service], [usage])};
- 4\*3 Member: four members in each dimension, e.g., {[service], ([minute], [text], [data], [web])};
- 3 Levels of time dimension: year, season, and month, e.g., {[time], ([month], [week], [day], [minute])};
- 16\*4 Sets: a BI cube is a BI set, e.g., {[time, service, usage]}.



**Figure 1: A Business Intelligence Base (BIB) of Online Mobile Retailing System**

Each business intelligence cube contains two sets of BI information: BI concepts and relationships of BI concepts. As it can be imaged, the entire BIB is multi-dimensional. In Figure 1, there are approximately 128 sub-dimensions for service-consuming analysis. Thus, it uncovers the first task in BIR: BIB decomposition, see Section 2.

The paper is organised as follows: Section 2 introduces the decomposition method to partition BIB into business intelligence slices; Section 3 introduces a programming style-based program partitioning method to partition program code into function-oriented program modules; Section 4 introduces BI concept recovery method; Section 5 introduces BI concept analysis to model recovered BI results; Section 6 presents a case study of Auto-BIR in an online mobile retailing system; Section 7 is related work; and Section 8 is conclusion.

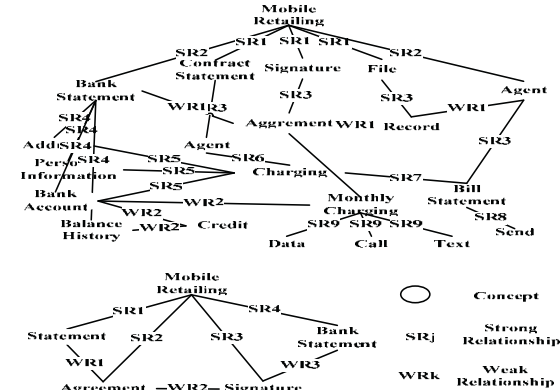
## 2. Partitioning Business Intelligence Base into Business Intelligence Slices

The idea of partitioning business intelligence base into business intelligence slices and using the business intelligence slices to recovery business intelligence from source code is in accordance with Finite Element method [3] commonly used in mechanical engineering fields. In Finite Element method, a finite number of basic structures, such as triangles and circles, are used to approximate more complex shapes in the real world.

Since the unnecessary relationships between business intelligence concepts are removed as the business intelligence base is partitioned into business intelligence slices, the size of overall business intelligence base is significantly reduced. This means that it can get a cost-effective business intelligence base and benefits will also be gained when recovering business intelligence from source code using this business intelligence base.

Figure 2 presents an example of simplifying BI of “Mobile-Plan Agreement and Assignment” to a piece of business intelligence slice of “contract agreement” on

base of Strong Relationship (SR) and Weak Relationship (WR) about a concept: “file”.



**Figure 2: Simplification of a BI of “Mobile-Plan Agreement and Assignment”**

In Figure 2, business intelligence slice of “contract agreement” is proposed as a cube of business intelligence base. The “simplification” algorithm is not presented since business intelligence base partition can be done offline and it is not of the top research priority.

## 3. Business Intelligence Oriented Program Partitioning

Following Section 2, Section 3 introduces a proposed method of program partitioning. In BIR, a good match between program modules and business intelligence slices is one of the successful criteria in research. Therefore, this part of research needs to pay more attention.

Let  $P$  be a source program;  $PM_1, \dots, PM_n$  be a set of program modules that are the result of partitioning of  $P$ , each  $PM_i, 1 \leq i \leq n$  contains independent business intelligence. Let  $BIS$  be a Business Intelligence Slice;  $BIS_1, \dots, BIS_m$  be a set of partitions of  $BIB$ , each  $BIS_j, 1 \leq j \leq m$  contains independent sub-business intelligence from each other.

This program partitioning method generally involves three steps:

1. Find all the possible  $BIS$ s
2. Compute the coupling and coherence of each proposed  $BIS$
3. Choose program modules that have good value of  $BIS$  coupling and coherence

As it can be imaged, the computational effort of BI oriented program partitioning is extremely large. In order to reduce the computational effort, a new heuristic rule-based method has been sought to build efficient program partitioning algorithms: programming style based program partitioning.

It is identified that three groups of feathers in source code can be used to distinguish different programming



styles. They are style of comments, style of names and style of indentations.

Let  $PS_1, \dots, PS_m$  be  $m$  different groups of programming styles; let  $ps_{i,j}, 1 \leq i \leq m, 1 \leq j \leq nps_i$  be the different programming styles within group  $i$ , where  $nps_i$  is the total number of programming styles in group  $i$ . If only one programming style  $ps_{i,j}$  in one group  $PS_i$  is

<b>Comments</b>	Style 1:        /* X */ Style 2:        // X Style 3:        /* *X ... */ Style 4:        /* X ... */
<b>Naming Conventions</b>	Style 1:        Connection-Mode Style 2:        Connection_Mode Style 3:        ConnectionMode
<b>Program Indentation</b>	Style 1:        bX Style 2:        bbX Style n:        b...bX

**Table 1: Programming Styles**

allowed to use to distinguish business intelligence slices, it will have totally  $nps_1 \times \dots \times nps_m$  different signatures to distinguish business intelligence slices.

Theoretically, if all the possible groups of programming styles and all the possible programming styles within each group are listed, business intelligence slices will virtually be able to distinguish from source program. In practice, some commonly-used groups of programming styles will sufficient to partition a source program into reasonable small program modules, see Table 1.

A program will go through a partitioning process which mainly has two stages, namely, programming style sampling, and program cutting, see Script 1, Script 2 for algorithms of program partition and Note.

#### 4. Business Intelligence Concept Recovery

Given a source program, the major clue for the existence of business intelligence slice is the names of variable types and procedures and certain program constructs implying business intelligence relationships. However, the names embedded in source program often occur as abbreviations and interpreting certain program constructs as some business intelligence oriented relationships may not always be appropriate, which makes business intelligence embedded in the source program ambiguous.

The first step at Section 4 is to collect all the procedure names and variable type names from the program module.

Once this has been done, concept recovery rules can then be applied to each of these names to recovery

business intelligence concepts. These concept recovery rules are classified into direct matching, regular atomic name recovery rules, irregular atomic name recovery rules, regular compound name recovery rule and irregular compound name recovery rules. The belief of a perfect match is 1, see Table 2.

```

PS ← null
CPL ← 1
SF ← null
WHILE CPL <> END-OF-PROGRAM DO
  PS ← programming style in CPL
  IF ps in PS THEN
    PS[ps] ← PS[ps] + 1
  ELSE
    PS ≤= ps
    PS[ps] ← 1
  ENDIF

  IF (CPL mod SI) == 0 THEN
    SF ≤= PS
    PS ← null
  ENDIF

  CPL ← CPL + 1
ENDWHILE

```

#### Script 1: Program Style Sampling Algorithm

```

Sp ← 1
CPs ← null
WHILE Sp <> SN DO
  IF ( SF [Sp] [PSSC] <> SF [Sp+1] [PSSC] ) OR
     ( SF [Sp] [PSVN] <> SF [Sp+1] [PSVN] ) OR
     ( SF [Sp] [PSUI] <> SF [Sp+1] [PSUI] ) OR
     ( SF [Sp] [PSMC] <> SF [Sp+1] [PSMC] ) OR
     ( SF [Sp] [PSPN] <> SF [Sp+1] [PSPN] ) OR
     ( SF [Sp] [PSNI] <> SF [Sp+1] [PSNI] )
  THEN
    CPs ≤= Sp
  ENDIF
  Sp ← Sp + 1
ENDWHILE

```

#### Script 2: Program Cutting Algorithm

##### Note:

- ← stands for the operation of assignment
- ≤= stands for the operation of adding an element to a set
- Programming Style (PS)  
Current Program Line (CPL)  
Sampling Function (SF)  
Sample Interval (SI)  
Pointer of Sampling Function (Sp)  
Name of Samples (SN)  
Cutting Points (CPs)  
Programming Style for Single-line Comment (PSSC)  
Programming Style for Multiple-line Comment (PSMC)  
Programming Style for Naming of Variable (PSVN)  
Programming Style for Naming of Procedure (PSPN)  
Programming Style for Un-nest Indentation (PSUI)

	Rule	Example	Belief
Direct Matching	Completely match	address (address)	1
Regular Atomic Name	First five letters	promo (promotion)	0.5
Irregular Atomic Name	One to one	stmtnt (statement)	0.7
Regular Compound Name	Completely Match	email_push ( $\emptyset$ )	0
Irregular Compound Name	Mix Match	fname (first name)	0.9

**Table 2: Business Intelligence Concept Recovery**

Once the concepts in a program module have been recovered, the relationship between these concepts must be generated also. This is done in formal concept analysis model where formal concept lattice is used to extract relationships from program constructs such as a sequences of procedures calls, a single procedure call, construct definition, database schema, etc. Since program constructs can indicate relationship at only structural level, these recovered relationships are named raw relationships. This part of research has been discussed in past works. Interested reviewer could refer to [10, 11, 12, 18, 19, 20].

### 5. Business Intelligence Modelling via Formal Concept Analysis (FCA)

Let us call a tuple of sets,  $(G, M, I)$ , that verify  $I \subseteq (G \times M)$ , formal context.  $G$  is usually called a set of objects and  $M$  a set of attributes. The binary relation  $I$  gives the incidence of the set of attributes on the set of objects. It is then possible to define the following applications, in whose definitions the notions, respectively, of the set of attributes that certain objects possess, and the set of objects that certain attributes possess, can be seen:

$$\begin{aligned} \varphi: \wp(M) &\rightarrow \wp(G) \\ A \mapsto A^\uparrow &= \{m \in M \mid (g, m) \in I, \forall g \in A\} \\ \psi: \wp(G) &\rightarrow \wp(M) \\ B \mapsto B^\downarrow &= \{g \in G \mid (g, m) \in I, \forall m \in B\} \end{aligned}$$

These two definitions allow the following definition to be made, that reflects the informal notion of concept as a set of objects and attributes that are mutually determined.

Definition: Let us call a pair  $(A, B) \in \wp(M) \times \wp(G)$  that verifies  $A^\uparrow = B$  and  $B^\downarrow = A$ , a formal concept. Normally, the first set in the pair will be called the concept extent and the second, the concept intent. The set of formal concepts associated to a context  $(G, M, I)$  will be denoted as  $G(G, M, I)$ .

On  $G(G, M, I)$  a partial order relation can be defined through the following formula where  $(A, B), (A', B') \in G(G, M, I)$ :

$$(A, B) \leq (A', B') \Leftrightarrow A \subseteq A' (\Leftrightarrow B \supseteq B')$$

From this definition, the following result can now be proved. Theorem (Fundamental for concept lattice): The set  $G(G, M, I)$  with the defined partial order relation forms a complete lattice in which the lowest and highest are given by the following formulas where  $T$  denotes a set of indices, not necessarily finite, and  $\forall t \in T, (A_t, B_t) \in G(G, M, I)$ :

$$\begin{aligned} \bigwedge_{t \in T} (A_t, B_t) &= \left( \bigcap_{t \in T} A_t, \left( \bigcup_{t \in T} B_t \right)^\uparrow \right) \\ \bigvee_{t \in T} (A_t, B_t) &= \left( \left( \bigcup_{t \in T} A_t \right)^\downarrow, \bigcap_{t \in T} B_t \right) \end{aligned}$$

The existence of the lowest and highest for any set of concepts allows the following functions to be defined:

$$\begin{aligned} \gamma: G &\rightarrow G(G, M, I) \\ g &\mapsto \bigwedge_{\{(A, B) \in G(G, M, I) \mid g \in A\}} (A, B) \\ \mu: M &\rightarrow G(G, M, I) \\ m &\mapsto \bigvee_{\{(A, B) \in G(G, M, I) \mid m \in B\}} (A, B) \end{aligned}$$

It is easy to show that these functions admit a much simpler notation, as follows:

$$\begin{aligned} \forall g \in G, \gamma(g) &= (g^\uparrow, g^\uparrow) \\ \forall m \in M, \mu(m) &= (m^\downarrow, m^\downarrow) \end{aligned}$$

This provides a practical way of determining the largest concept in whose extent a certain object appears, or which other objects share all the attributes of a given object. Each node in such diagram represents a formal concept and each arc indicates an order relation between two concepts, where the larger is place above the smaller, with the restriction that no intermediate concept exists.

### 6. Case Study

An online mobile retailing system is chosen as case study in this paper. The 150K LOC is only partly shown. Steps of the case study follow the four procedures in BIR which have been mentioned above. ‘‘Business Intelligence Slices’’ oriented ‘‘Program Modules’’ are presented in code segments, see Script 3. An overall result of BIR, in addition, is modelling and presented via formal concept analysis in Table 3 and Figure 3.

```
ServiceConsumingAnalysis {
...
  {ConsumingAnalysis_2009.01 = Consuming_Analysis_Month;
  {Service_id = Service ID;}
  ...
}
...
_freeService {
  if ((Data = null)) {
    Data Close();
  }
  if ((Minute = null)) {
    Minute Close();
  }
  ...
}
```

```

...
Service_ID {
  lterm_No.,
  Minute,
  Text,
  Data,
  ...
};
...
{TIME_level
  Month = MONTH;
  ...
}
...
Service_ID {
  _Minutes;
  _Text;
  _Data;
  _Web
};
...
Data_ID {
  _Email = ();
  _Wi-Fi = ();
  ...
};
...

```

**Script 3: Code Segement of BIR Case Study**

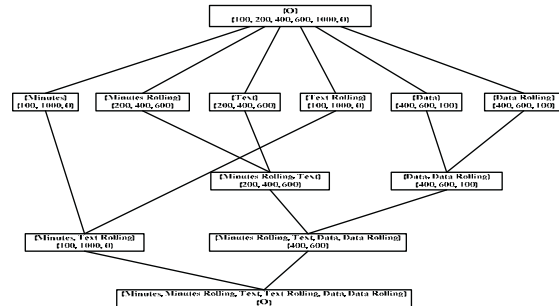
The tabular context of business intelligence “plan rolling” in the application is a specific module to handle service calculation and control service rolling in accordance with the mobile contract and the status of service consuming. Table 3 could generate a piece of business intelligence. It looks like those BI concepts and data have been gotten, but it is abstract to understand. As mentioned, engineers are lacking of business oriented program comprehensibility when being front of a large amount of data and code. A better idea is building BI lattice with those recovered BI information.

	Min utes	FreeMinu tesRolling	Tex t	FeeText Rolling	Data	FreeDat aRolling
100	V			V		
200		V	V			
400		V	V		V	V
600		V	V		V	V
1000	V			V	V	V
0	V			V		

**Table 3: Tabular Context of “PlanRolling”**

As seen in Figure 3, mobile plans 400 and 600 deal with minutes rolling and data rolling quite often. It makes sense since both plans quote 400 and 600 minutes with 15 GB internet fair usage every month. But for the plans 100, 1000 and 0, the situation is different; plans 100 and 0 do not quote any free internet services, while plan 1000 is designed for the premier customers who are willing to pay 75 GBP every month for 1000 minutes, 3000 text and unlimited data. The recovered business intelligence slice

matches perfectly to the business information in program code.



**Figure 3: Concept Lattice of “PlanRolling”**

## 7. Related Work

With the business idea of Web, computing platform is evolving at a more than predicable speed. It brings a significantly difficult and remarkable task to software engineers: understanding the computing completely.

Foundations of reverse engineering [4, 17] and software evolution [13, 1] have been established since 1990s by groups of researchers. In those days, it was very crucial to have a sound reverse engineering and evolution methods to comprehend program code, identify reusable components, and evolve the system architecture, etc.

Soon after, reverse engineering researchers turned more attention to semi-functional information embedded in a software system, e.g., program transformation [21], program abstraction [15], domain knowledge recovery [14], business rule extraction [7], etc.

Today, the phenomenon is continuously evolving, along with the growing demand to meet business dynamics, system is operated one upgrade after another. It is because computing system needs to be fully taken advantage from in daily business to keep itself in the leading position in business world. This idea is not easily achievable for a certain number of software systems. One of the causes is the automatic mechanism of business intelligence recovery is not guaranteed for those systems.

Therefore, data mining and knowledge recovery [5, 2, 6] methodologies continuously keep their promises in accessing information in software analysis [22, 23] and software evolution [24].

## 8. Summary

Generally, on one hand, automatic reverse engineering is bottlenecked with lacking program comprehensibility. The software mining work is not complete until program comprehensibility is fully obtainable from program code. Ideally, a business intelligence base is one of the sources

for program comprehensibility. The successful recovery of business intelligence and its matches with program code are meaningful to reverse engineering works. On the other hand, business information system, especially a system running in Web-based computing environment, needs automating business intelligence recovery method to explore the business data and logic.

Specifically, this paper presents an automatic path to recover business intelligence from software system. It is a software reverse engineering process of four procedures: BIB decomposition, BI oriented program partitioning, BI concept recovery, and BI formal concept analysis. Each procedure is a sub-task in Auto-BIR.

The future research work is viewed as “automating business intelligence accumulation in Web” to bridge this work to nowadays Web computing environment.

## References

- [1] K. H. Bennett and V. T. Rajlich, “Software Maintenance and Evolution: A Roadmap”, In *Proceedings of the 22nd IEEE/ACM International Conference on Software Engineering (ICSE'00)*, pp. 73-87, Limerick, Ireland, June 2000.
- [2] T. J. Biggerstaff, B. G. Mitbander and D. Webster, “The Concept Assignment Problem in Program Understanding”, In *Proceedings of 15th IEEE/ACM International Conference on Software Engineering (ICSE'93)*, pp. 482-498, Los Alamitos, CA, USA, April 1993.
- [3] D. Burnett, *Finite Element Analysis: from Concepts to Applications*, Addison-Wesley Publican Co., October 1987.
- [4] E. Chikofsky and J. Cross, “Reverse Engineering and Design Recovery: A Taxonomy”, *IEEE Software*, vol. 7, no. 1, pp. 13-17, January 1990.
- [5] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, “From Data Mining to Knowledge Discover in Databases”, *American Association for Artificial Intelligence (AAAI'96)*, vol. 17, no. 3, pp. 37-54, 1996.
- [6] W. B. Frakes and K. Kyo, “Software Reuse Research: Status and Future”, *IEEE Transactions on Software Engineering (TSE'05)*, vol. 31, no. 7, pp. 529-536, July 2005.
- [7] H. Huang, W. Tsai, S. Bhattacharya, X. Chen, Y. Wang and J. Sun, “Business Rule Extraction from Legacy Code”, In *Proceedings of the 20th IEEE Conference on Computer Software and Applications (COMPSAC'96)*, pp. 162-167, 1996.
- [8] J. Kang, *Automating Business Intelligence Recovery in Software Evolution*, Ph.D. Thesis, De Montfort University, 2009. (submitted)
- [9] J. Kang, J. Pu, J. Huang, Z. Zhou and H. Yang “Business Intelligence Recovery in Reverse Engineering”, In *Proceedings of the 2nd IEEE International Workshop on Quality Oriented Reuse of Software (QUORS'08)*, COMPSAC Workshops, pp. 765-770, Turku, Finland, July 2008.
- [10] J. Kang and H. Yang, “Supporting Static and Dynamic Feature Modelling by Formal Concept Analysis”, In *Pre-Proceedings of the IEEE International Workshop on Software Technology and Engineering Practice (STEP'05)*, pp. 133-135, ICSM Workshops, Budapest, Hungary, September 2005.
- [11] J. Kang and H. Yang, “Modelling Web Applications via Formal Concept Analysis”, In *Proceedings of the 11th Chinese Automation and Computer Society in UK (CACISUK'05)*, Sheffield, UK, August 2005.
- [12] J. Kang, H. Zhou and H. Yang, “Task Decomposition for Communication Computation Overlap to Reengineer a Web-Based System”, In *Proceedings of the 11th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'07)*, pp. 205-212, Sedona, Arizona, USA, March 2007.
- [13] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry and W. M. Turski, “Metrics and Laws of Software Evolution – The Nineties View”, In *Proceedings of the 4th IEEE International Software Metrics Symposium (METRICS'97)*, pp. 20-32, 1997.
- [14] Y. Li, *Automating Domain Knowledge Recovery from Legacy Code*, Ph.D. Thesis, De Montfort University, 2002.
- [15] X. Liu, *Abstracting: A Notion for Reverse Engineering*, Ph.D. Thesis, De Montfort University, 1999.
- [16] H. P. Luhn “A Business Intelligence System”, *IBM Journal*, October 1958.
- [17] H. A. Muller, J. H. Jahnke, D. B. Smith, M. Storey, S. R. Tilley and K. Wong, “Reverse Engineering: A Roadmap”, In *Proceedings of the 22nd IEEE/ACM International Conference on Software Engineering (ICSE'00)*, pp. 47-60, Limerick, Ireland, June 2000.
- [18] J. Pu, Z. Zhang, J. Kang, Y. Xu and H. Yang, “Using Aspect Orientation in Understanding Legacy COBOL Code”, In *Proceedings of the 1st IEEE International Workshop on Quality Oriented Reuse of Software (QUORS'07)*, COMPSAC Workshops, pp. 385-390, Beijing, China, July 2007.
- [19] Z. Zhang, J. Kang and H. Yang, “UML Modelling Web Applications via Formal Concept Analysis”, In *Proceedings of the 18th KSI International Conference on Software Engineering and Knowledge Engineering (SEKE'06)*, pp. 532-535, San Francisco, CA, USA, July 2006.
- [20] H. Zhou, J. Kang and H. Yang “OPTIMA: An Ontology-Based Platform-Specific Software Migration Approach”, In *Proceedings of the 7th IEEE International Conference on Quality Software (QSIC'07)*, pp. 143-152, Portland, OR, USA, October 2007.
- [21] M. Ward, *Proving Program Refinements and Transformations*, Ph.D. Thesis, Oxford University, 1989.
- [22] M. Weiser, “Program Slicing”, *IEEE Transactions on Software Engineering (TSE'84)*, vol. 10, no. 4, pp. 352-357, July 1984.
- [23] R. Wille, *Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts*, Dordrecht-Boston, March 1982.
- [24] H. Yang and M. Ward, *Successful Evolution of Software Systems*, Artech House Publishers, January 2003.

# Automatic Class Matching to Compare Extracted Class Diagrams: Approach and Case Study

Yan Liang, Nicholas A. Kraft, and Randy K. Smith  
*Department of Computer Science*  
*The University of Alabama*  
*Tuscaloosa, Alabama 35487-0290, USA*  
{yliang, nkraft, rsmith}@cs.ua.edu

## Abstract

*Reverse engineering tools often are employed by software maintenance teams. The abundance of these tools makes evaluation and cross-tool comparisons a difficult and time-consuming process. Ignored by most evaluation approaches and experiments is the understanding of similarities and differences of the output artifacts produced by different tools. We propose a novel approach to facilitate the tool evaluation process by examining output artifacts directly. We present a class matching algorithm to automatically detect whether a given pair of classes represent the same entity in source code. The algorithm divides all classes extracted from each candidate tool into three categories: matched, unmatched, and unknown. In this case study, we demonstrate this approach by comparing the class diagrams for four C++ projects extracted by two open source tools: Doxygen and StarUML. In a follow-up qualitative analysis of the matching results, we evaluate the matching precision of our approach and its capacity to reveal the differences between analysis capabilities of different class diagram extractors.*

## 1. Introduction

Program comprehension is a critical task for maintainers of legacy systems [1]. Reverse engineering (RE) tools simplify the program comprehension process by automatically generating textual and graphical reports of design, dependencies, and code structure. Call graphs and UML class diagrams are examples of commonly extracted artifacts.

Different RE tools, even with similar technology, extract artifacts that vary dramatically. There are several reasons for the inconsistency of the output artifacts. Many RE tools use a compiler front end to build abstract syntax trees (ASTs) and then extract information from those ASTs. Other tools relax the AST construction rules as a cost-efficiency measure and to improve comprehensibility for the average developer. Additionally, the intricacies of different programming languages lead to a lack of widely accepted correspondence between language elements and artifact constructs [2]. Lastly, although standard exchange formats such as XML Metadata Interchange (XMI) [3] and the Graph

eXchange Language (GXL) [4] have been adopted by many RE tools, tool interoperability is far from satisfactory [5]. Therefore, software developers need an objective tool evaluation approach to compare competing tools and identify the best tool for a given task.

## 2. Motivation

A common approach to tool evaluation is to evaluate output precision, which is typically measured by mapping derived artifacts back to corresponding entities from the input documents. This work commonly is done by hand due to the lack of such mapping functions embedded in the tools or the inconsistency of mapping rules among tools [6], [7]. We seek to improve on this time-consuming process by investigating output-to-output mapping for RE tools. Our approach addresses two questions:

- Is it feasible to automatically identify two entities (in our case – classes) in the output artifacts produced by two tools that correspond to the same entity in the input documents?
- Is this approach an effective way to examine the similarities and disparities between tools?

Research on record matching, or record linkage, guides us on how to identify two entities with different representations but the same meaning [8], [9]. We use domain knowledge of record matching and reverse engineering, along with similarity metrics, to partition classes extracted by two tools into three disjoint sets: match, unmatched, and unknown. We believe that such automatic classification will provide quickly the qualitative evaluation of the tools needed by developers and maintenance teams.

## 3. Class Matching Algorithm

In this section, we provide the details of our class matching algorithm for pairwise comparison of class diagrams produced by competing RE tools. For each class extracted by a tool, the algorithm determines whether there exists a corresponding class extracted by the other tool. We first describe the representation of a class diagram on which the algorithm operates and then describe the foundations of the algorithm, including the encountered challenges.

### 3.1. Representation of Class Diagrams

Each class,  $c$ , in an extracted class diagram,  $D$ , has three components: name, attribute set, and operation set. We currently do not consider other components, such as relationships, that may be extracted by a tool,  $T$ . We use the term *class* to indicate any record data type, such as a C++ class, struct, or union.

A name, such as a class or type name, is represented as a string. An attribute can be represented by its name or by a two-dimensional vector containing its name and type name. An operation can be represented by its name or by an  $n$ -dimensional vector containing its name, return type name, and parameter type name sequence. We use the phrase *parameter type name sequence* rather than *parameter type name set* to indicate ordering. However, we use the term *set* for attributes and operations. A tool may extract them in a particular order, but our algorithm ignores order in these cases. Further, if an operation is represented only by its name, all extracted class operations comprise a multiset (bag). In this case, the multiplicity of a name within the multiset corresponds to the concept of method overloading in object-oriented languages.

Suppose that we have two class diagrams  $D_1$  and  $D_2$  extracted by two RE tools,  $T_1$  and  $T_2$ , respectively, for a single input project. Based on the class matching model in our previous work [10],  $D_1$  and  $D_2$  containing  $P$  and  $Q$  classes, respectively, are represented as follows:

$$D_1 = \{c_i = (clsName_i, attrSet_i, operSet_i) : i \in [1, P]\}$$

$$D_2 = \{c_j = (clsName_j, attrSet_j, operSet_j) : j \in [1, Q]\}$$

### 3.2. Similarity of Class Pairs

Looking at individual classes in the class sets, the similarity score of a class pair,  $sim(c_i, c_j)$ , is indicative of how close two classes are, and is computed as the average of the similarities of the three components – each of which is between 0 and 1.

$$Sim_{clsName} = sim(c_i.clsName, c_j.clsName)$$

$$Sim_{attrSet} = sim(c_i.attrSet, c_j.attrSet)$$

$$Sim_{operSet} = sim(c_i.operSet, c_j.operSet)$$

$$sim(c_i, c_j) = \frac{Sim_{clsName} + Sim_{attrSet} + Sim_{operSet}}{3}$$

We describe computation of similarity scores for class names in the following subsection. The similarity score for two attribute sets is computed by:

$$sim(c_i.attrSet, c_j.attrSet) = \frac{2|c_i.attrSet \cap c_j.attrSet|}{|c_i.attrSet| + |c_j.attrSet|}$$

The similarity score for two operation sets is computed analogously, and the similarity score for empty sets is zero.

### 3.3. Similarity of Class Names

Class name matching is not a trivial exercise. For C++ an RE tool can extract a fully-qualified name from source code using only syntactic information, but cannot always guarantee completeness/correctness due to the presence of the preprocessor or the use of a fuzzy parser [11]. The consequence is that, even if many RE tools adopt the naming rule of class pathname in the form of `namespace::clsName` for a UML class diagram, tools may extract different names for a single entity in the source code. For example, `Jikes::Annotation::Component` and `Annotation::Component` probably refer to the same class in the source code.

To address the class name matching problem, in our approach we measure the similarity between two name strings. A name string is a sequence of case sensitive terms separated by a delimiter. Computing the similarity of two class name strings has three cases:

- Case 1:** If the class name strings are exactly the same, which means they have equal length and the same terms in the same order, their similarity score is 1.
- Case 2:** If the class name strings have no common terms, their similarity score is 0.
- Case 3:** If the class name strings are not exactly the same but have terms shared, we use a token-based string matching algorithm to compute the cosine similarity of the name strings. We use the tf-idf (term frequency-inverse document frequency) term weighting scheme, a name matching method first proposed by Cohen for the WHIRL system [12].

The first two cases are straightforward. The third case requires further consideration. For tf-idf, a term  $t$  of a name string  $n$  is assigned a weight computed by:

$$w_n(t) = \log(tf_{t,n} + 1) * \log(idf_t)$$

$tf_{t,n}$  is the number of occurrences of  $t$  in the name string  $n$ , and  $idf_t$  is the value obtained by dividing the number of all class names from two class diagrams for a single project by the number of those names containing the term  $t$ .

The cosine similarity between two names  $n_1$  and  $n_2$  is:

$$sim(n_1, n_2) = \frac{\sum_{k=1}^R w_{n_1}(t_k) * w_{n_2}(t_k)}{\sqrt{\sum_{k=1}^R (w_{n_1}(t_k))^2 * \sum_{k=1}^R (w_{n_2}(t_k))^2}}$$

$R$  is the number of distinct terms in name strings  $n_1$  and  $n_2$ .  $w_{n_1}(t_k)$  and  $w_{n_2}(t_k)$  are the weights of term  $t_k$  in  $n_1$  and  $n_2$ , respectively. The cosine similarity metric is insensitive to the location of term, which is critical for those name strings with terms missed by the extraction algorithm.

In some cases, a class name similarity score only may be used to determine an overall similarity score for two classes. See Section 3.5 for details.

### 3.4. Classification of Extracted Classes

Class pair similarity scores are the foundation of our scheme for pairwise comparison of extracted class diagrams. Recall that we are guided by the approach of Fellegi and Sunter [8], who classify pairs into three categories. Thus, we define a classifier,  $f$ , that determines into which category an extracted class should be placed: match set, unmatched set, and unknown set. The classifier  $f$  is built upon a function,  $h$ . For a class  $c$  extracted by one tool,  $h(c)$  finds the class(es) extracted by the other tool that is most similar to  $c$ .

For clarity we define  $f$ , and the three predicates that form it, using symbols from Section 3.1. In particular, assume we are classifying the classes of class diagram  $D_1$  and that  $c_i$  is the extracted class from  $D_1$  currently being considered.

$$match = \exists c_j \in D_2 : h(c_i) = c_j \wedge h(c_j) = c_i$$

$$unmatch = \forall c_j \in D_2 : sim(c_i, c_j) = 0$$

$$unknown = \nexists c_j \in D_2 : h(c_i) = c_j \wedge h(c_j) = c_i$$

$$f(c_i) = \begin{cases} c_i \in matchSet_1 & \text{if } match \text{ is true;} \\ c_i \in unmatchSet_1 & \text{if } unmatch \text{ is true;} \\ c_i \in unknownSet_1 & \text{if } unknown \text{ is true.} \end{cases}$$

The predicate *match* is true for a class  $c_i \in D_1$  when  $h(c_i)$  returns exactly one class  $c_j \in D_2$  and  $h(c_j)$  returns only  $c_i$ . That is, among all class pairs involving  $c_i$  or  $c_j$ ,  $(c_i, c_j)$  is the class pair with the single highest similarity score. We call such a match a *stable match*, and in the case of a stable match for a class pair  $(c_i, c_j)$  we place  $c_i$  in  $matchSet_1$  and  $c_j$  in  $matchSet_2$ .

The predicate *unmatch* is true for a class  $c_i \in D_1$  when, for every class  $c_j \in D_2$ , the similarity score of  $(c_i, c_j)$  is zero. In such a case, we place  $c_i$  in  $unmatchSet_1$ . Therefore, if a class is extracted by one of the tools being compared, but not the other, the class is placed in the appropriate unmatch set.

The predicate *unknown* is true for a class  $c_i \in D_1$  when there does not exist a class  $c_j \in D_2$  for which  $(c_i, c_j)$  is the class pair with the single highest similarity score among all class pairs involving  $c_i$  or  $c_j$ . That is, either (or both) of  $h(c_i)$  and  $h(c_j)$  returns more than one class. We call such a match an *unstable match*, and in the case of an unstable match for a class pair  $(c_i, c_j)$  we place  $c_i$  in  $unknownSet_1$  and  $c_j$  in  $unknownSet_2$ .

Using our matching classifier,  $f$ , we partition all extracted classes for a class diagram into one of our three categories: match set, unmatch set, and unknown set. These three sets are collectively exhaustive and mutually exclusive: their union includes all extracted classes for the class diagram and their intersection is the empty set.

### 3.5. A Generalized Approach

We now have described the constituent parts of our class matching algorithm. Figure 1 illustrates the entire algorithm

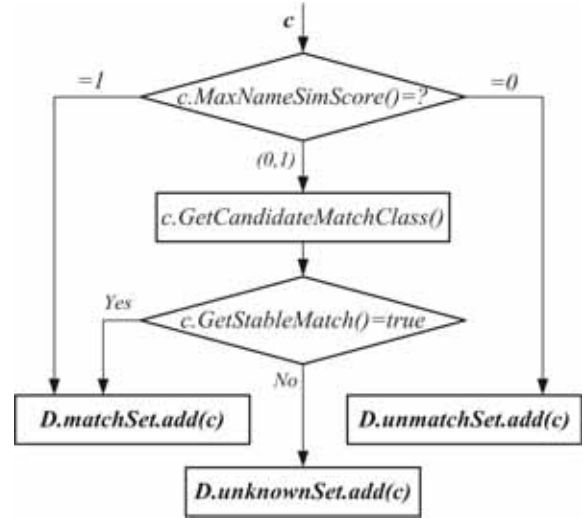


Figure 1. Algorithm Control Flow

as a flow chart. For each extracted class  $c$  from class diagram  $D$ , we first attempt to determine the overall class similarity score using only the class name. If there exists an extracted class in the other class diagram whose name is exactly the same as that of  $c$  then  $c$  is added to the *matchSet* for  $D$ . If there exists no extracted class in the other class diagram whose name shares common terms with that of  $c$  then  $c$  is added to the *unmatchSet* for  $D$ . If necessary, we find the class(es) in the other class diagram that has the highest similarity score with  $c$ . If we find a stable match, we add  $c$  to the *matchSet* for  $D$ ; otherwise, we add  $c$  to the *unknownSet* for  $D$ . When our class matching algorithm terminates, each extracted class from a class diagram belongs to exactly one of the three sets for that class diagram.

## 4. Experiment

The two class diagram extractors we study are Doxygen 1.4.4 [13] and StarUML 5.0 [14]. The output of Doxygen is a custom XML format, and the output of StarUML conforms to XMI 1.1 and UML 1.3. Our first step toward class matching is to perform a series of semantics-preserving data transformations on the output of each tool. After applying these transformations, we obtain sets of CSV files that encode the extracted class diagrams. The implementation of our class matching algorithm accepts these files as input.

Table 1 lists summary information for the four open source C++ projects in our test suite. FOX is a toolkit for graphical user interface development [15]. Jikes is a Java compiler system from IBM [16]. Pixie is a RenderMan® like photorealistic renderer [17]. Scintilla is a source code editing component [18].

### 4.1. Class Matching Results

Table 2 lists the sets computed by our class matching algorithm. The value in each cell is the number of classes

Project	Version	LOC ( $\approx$ K)	Classes Extracted by Doxygen	Classes Extracted by StarUML
FOX	1.4.17	110	289	123
Jikes	1.22	70	275	283
Pixie	1.5.2	80	254	220
Scintilla	1.66	35	93	78

Table 1. Test Suite for Experiment

Project	Tool	Unmatch Set	Match Set		Unknown Set
			exact	similar	
FOX	Doxygen	2	116	3	168
	StarUML	3	116	3	1
Jikes	Doxygen	3	5	267	0
	StarUML	3	5	267	8
Pixie	Doxygen	52	200	2	0
	StarUML	12	200	2	6
Scintilla	Doxygen	19	74	0	0
	StarUML	1	74	0	3

Table 2. Class Matching Results. *The last four columns list the numbers of classes in the corresponding sets.*

in the indicated set for the indicated project and tool. There are two cases for match set (exact or similar), and we report these separately in the table. For each project, the sizes of the match sets for the two tools are always equal due to the constraint provided by stable matching. Recall from Section 3.1 that attributes and operations may be represented in multiple ways; for this experiment we use the name-only representations.

From Table 2, we observe:

- For the three projects other than FOX, most of the extracted classes are placed into the match set. For example, for Scintilla 74 of 93 classes extracted by Doxygen and 74 of 78 classes extracted by StarUML are exact matches.
- For the three projects other than Jikes, most of the classes in the match set are exact (stable) matches. For Jikes, most of the classes in the match set have names similar to their peers.
- Doxygen and StarUML each yield a number of unmatch classes for each project.
- StarUML extracts unknown classes for each of the four projects; in contrast, Doxygen extracts unknown classes only for FOX. In particular, over 50% of the classes extracted by Doxygen for FOX are unknown classes.

## 4.2. Analysis of Experimental Results

Our analysis centers on the two questions proposed in Section 2: Can our approach match and classify classes correctly? and Is such a classification an effective way to analyze features between tools? To this end, we gave four volunteers a project, both source code and class diagrams obtained using Doxygen and StarUML. Their task was to validate manually each extracted class in the source code. A comparison between each extracted class and its

corresponding location in the source code is the only way to tell us not only the quality of classification, but also the potential causality between language syntax structure and analysis features of a particular class diagram extractor. The correctness or quality of classification is measured by the number of false positives and false negatives.

### 4.2.1. Match Set

Manual matching results indicate that our class matching algorithm results in no false positives: every pair of extracted classes in the match sets are real matches, referring to the same class in the source code. This result includes both exact and similar matches. For instance, in Pixie, we successfully identify that `COptions::CDisplay::` extracted by StarUML matches `COptions::CDisplay::TDisplayParameter` extracted by Doxygen, and that `CShader::TShderParameter` matches `CShader::`. This is beyond our expectation because we intuitively believe an equally weighting scheme for the similarity calculation between classes could cause false positives. Matching precision is a qualitative measurement of a matching algorithm that is widely accepted by the research community of record matching. Zero occurrence of false positive means 100% matching precision for this case study where two tools and four projects are involved.

Most notable in the match set is the large amount of matches with similar class name for Jikes. Inspection on related code provides the answer: every Jikes source file uses the preprocessor to conditionally define the Jikes namespace. For all classes defined in these source files, Doxygen ignores this namespace and outputs the original class name, while StarUML imposes the namespace as the prefix on all extracted classes. As an example, consider the extracted a piece of code below from the header file `unzip.h` of Jikes. Doxygen extracts the class name `huft`, but StarUML extracts `Jikes::huft`.

```

#ifdef HAVE_JIKES_NAMESPACE
namespace Jikes {
#endif
struct huft {
    unsigned char e;
    unsigned char b;
    union {
        unsigned short n;
        struct huft *t;
    } v;
};

```

In practice, the attributes and operations for extracted classes placed in match sets based on name only may differ significantly. An extensive comparison of attributes and operations for each pair of matched classes is part of our future work.



### 4.2.2. Unmatch Set

Each class in an unmatch set is identified by our matching algorithm as being extracted by one tool but not by the other. The matching algorithm generates a few false negatives: some extracted classes with actual matches are improperly placed into this set. These false negatives result from one drawback of our matching algorithm: the comparison of class attributes and operations is not undertaken if two class name strings are found to have no similarity. For example, in Pixie, the matching algorithm misses the match between the class `TArgument` extracted by Doxygen and `_14` extracted by StarUML for the anonymous class shown in the following code. These name strings share no term, yet all of the class attributes are exactly the same. Ruling out this type of false negative is part of our future work.

```
typedef struct {
    unsigned char numArguments;
    unsigned char uniform;
    unsigned char numCodes;
    unsigned char plNumber;
} TArguments;
```

In further analysis, we examined those classes extracted by StarUML but not by Doxygen. Some of these extracted classes are conditionally defined using preprocessor directives. Given this finding and the preceding analysis of match sets, we infer that Doxygen and StarUML handle code with conditional compilation in different ways: StarUML ignores or is unaware of conditional compilation directives, while Doxygen ignores conditionally compiled code unless otherwise directed. Other unmatch classes extracted by StarUML have no attributes or operations and correspond to `extern` declarations in the source files. Overall, we postulate that StarUML trades extraction accuracy for performance gains, and important consideration for potential tool adopters.

### 4.2.3. Unknown Set

The unknown set covers those classes in a project where there exists a cognitive gap between manual matching and automatic matching. In the qualitative study, our subject matter experts find corresponding classes in source code for each of the extracted classes. However, this process does not always yield a one-to-one match, because a tool may (erroneously) extract multiple classes for a single class in source code. To the contrary, our matching algorithm forces a one-to-one match: the assumption is that a class defined in source code should be extracted as at most one class, so that one class extracted by one tool can match at most one class extracted by the other tool.

The large number of unknown classes extracted by Doxygen for the project FOX deserves careful analysis. We find that StarUML extracts a large class `FX::FXAPI<TYPE>` with a suspiciously large number of attributes, constructors, destructors, and other operations. Indeed, through our

manual analysis we discovered that many of these attributes and operations are actually members of other classes which StarUML did not extract. On the contrary, Doxygen correctly extracts those classes. We determined that a macro, `FXAPI`, is misinterpreted as a template class by StarUML.

Another area of interest is the unknown classes extracted by StarUML for Jikes, Pixie and Scintilla. The corresponding source code for most of these extracted classes have similar syntactic structure: they are attributes, but are deemed by StarUML to be a new classes due to the C-style use of keyword `struct` before the attribute type name. From the following code, StarUML extracts an class `SCNotification::NotifyHeader` without any attributes or operations. Such behavior indicates that StarUML uses lexical, rather syntactic, analysis to guide its reverse engineering process.

```
struct SCNotification {
    struct NotifyHeader nmhdr;
    int position;
    int ch;
    /* other data members not shown */
};
```

### 4.2.4. Summary

Our analysis shows that our matching algorithm generally performs well, but can be improved using the knowledge gained through this study. Similar issues for other reverse engineering tools have previously been discussed in the literature [7], [19], [20], but those issues either are not specific to class diagrams, or lack concrete details. We explore the behaviors of different class diagram extractors by classifying the extracted classes into small groups from which the details of discrepancies are easier to examine.

Generally, entities with missing data are not considered in the application of most matching algorithms. However, the use of such entities can not be avoided when performing output-to-output artifact matching among RE tools. Otherwise, the information that is most important to software developers or maintenance teams might be ignored. Our approach generally handles such missing information appropriately, as demonstrated by the low frequency of matching false positives and matching false negatives. A reliable matching algorithm that operates only on output artifacts benefits the tool evaluation process for its contribution of reducing evaluation effort, particularly the effort that would be otherwise spent mapping all derived entities back to the source code by hand.

### 4.3. Threats to Validity

There are threats to the internal and external validity of this study. One threat to internal validity is that the correctness of the manual matching results used in our qualitative analysis is dependent upon the knowledge of the human experts and their carefulness and patience when

reading source code. To avoid human errors, a second pass may be required to validate the initial findings. Threats to the external validity are our choices of tools and projects for the study. For example, because we paired Doxygen and StarUML with the Jikes project, we uncovered issues related to conditional compilation that can help us to improve our class matching algorithm. However, without experimenting with additional tools and projects, we can not be sure that there are not other issues that we have not yet uncovered.

## 5. Related Work

RE tool evaluation is an on-going issue in the literature. In this study we focus on evaluating textual outputs of tools, not on specific evaluation criteria such as precision metrics.

Murphy et al. [20] conducted an empirical study of static call graph extractors. They applied nine tools to extract lists of the calls between functions in C source code. Scripts were run on the output produced by the tools to transform the extracted call lists to the form `(function1;function2)` where `function1` calls `function2`. Call graphs were compared by computing the set intersection and difference of call sets. Details about the low-level comparison algorithm were not provided.

Sim et al. [7] designed a general-purpose benchmark to evaluate C++ fact extractors. The benchmark enumerates C++ language features, analysis problems, and reverse engineering issues as the basis for creating a task domain sample. Operators/expert users were then involved to check the accuracy of facts extracted.

A number of studies on tool evaluation compare the visualization capacities of tools [2], [21]. This approach can indirectly reflect the precision and richness of the data extracted by tools. Because what the user can see or would like to see is dependent on tool configuration and the downstream functionalities provided by tools, visualization-based approaches can not expose the entirety of similarities and differences among tools.

## 6. Conclusions and Future Work

In this paper we propose a class matching algorithm to automatically reveal the similarities and differences of tools by comparing their extracted output artifacts. The end result of our class matching algorithm is three categories of classes: match set, unmatched set and unknown set. Therefore, extracted classes are organized in such a way that the technical discrepancy between two extractors can be exposed in detail with relatively low effort.

As for our future work, we need examine more tools using more test cases to validate the applicability of our algorithm. Further, because the attributes and operations for extracted classes placed in match sets based on name only may differ significantly in practice, we need to incorporate comparison of attributes and operations for each pair of matched classes. The addition of such a comparison will

yield more operational information about the similarities and differences between the tools being compared. Finally, we need to revise our matching algorithm to eliminate false negatives that occur for anonymous classes when different tools use different naming schemes for such classes.

## References

- [1] H. Müller, J. Jahnke, D. Smith, M.-A. Storey, S. Tilley, and K. Wong, "Reverse engineering: A roadmap," in *Proceedings of the Future of Software Engineering*, Jun. 2000, pp. 47–60.
- [2] Y.-G. Guéhéneuc, "A systematic study of UML class diagram constituents for their abstract and precision recovery," in *Proc. of the 11<sup>th</sup> Asia-Pacific Software Engineering Conference*.
- [3] International Organization for Standardization, "Information Technology – XML Metadata Interchange," Geneva, Switzerland, Tech. Rep. ISO/IEC 19503:2005, 2005.
- [4] R. Holt, A. Schürr, S. E. Sim, and A. Winter, "GXL: A Graph-Based Standard Exchange Format for Reengineering," *Science of Computer Programming*, vol. 60, no. 4, pp. 149–170, 2006.
- [5] N. A. Kraft, B. A. Malloy, and J. F. Power, "An infrastructure to support interoperability in reverse engineering," *Informating and Software Technology*, vol. 49, no. 3, pp. 292–307, Mar. 2007.
- [6] I. T. Bowman, M. W. Godfrey, and R. C. Holt, "Connecting architecture reconstruction frameworks," *Information & Software Technology*, vol. 42, no. 2, pp. 91–102, 2000.
- [7] S. E. Sim, R. C. Holt, and S. Easterbrook, "On using a benchmark to evaluate C++ extractors," in *Proceedings of the 10<sup>th</sup> International Workshop on Program Comprehension*, Jun 26–29 2002, pp. 114–123.
- [8] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, Dec. 1969.
- [9] A. K. Elmagarmid, G. I. Panagiotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, Jan. 2007.
- [10] Y. Liang, "Automating matching artifacts for autonomous tool evaluation," in *Proceedings of the 47<sup>th</sup> ACM Southeast Conference*, Mar 19–21 2009.
- [11] N. A. Kraft, B. A. Malloy, and J. F. Power, "A tool chain for reverse engineering C++ applications," *Science of Computer Programming*, vol. 69, no. 1–3, pp. 3–13, Dec. 2007.
- [12] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *Proceedings of the ACM Workshop on Data Cleaning, Record Linkage and Object Identification*, Aug. 2003.
- [13] "Doxygen version 1.4.4," <http://stack.nl/dimitri/doxygen/>.
- [14] "StarUML version 5.0," <http://staruml.sourceforge.net/>.
- [15] "FOX Toolkit version 1.4.17," <http://www.fox-toolkit.org/>.
- [16] "Jikes version 1.22," <http://jikes.sourceforge.net/>.
- [17] "Pixie version 1.5.2," <http://pixie.sourceforge.net/>.
- [18] "Scintilla version 1.66," <http://www.scintilla.org/>.
- [19] M. N. Armstrong and C. Trudeau, "Evaluating architectural extractors," in *Proceedings of the 5<sup>th</sup> Working Conference on Reverse Engineering*, Oct 12–14 1998, pp. 30–39.
- [20] G. C. Murphy, D. Notkin, W. G. Griswold, and E. S. Lan, "An empirical study of static call graph extractors," *ACM Transactions on Software Engineering and Methodology*, vol. 7, no. 2, pp. 158–191, Apr. 1998.
- [21] S. Matzko, P. J. Clarke, T. H. Gibbs, B. A. Malloy, J. F. Power, and R. Monahan, "Reveal: A tool to reverse engineer class diagrams," in *Proceeding of 40<sup>th</sup> International Conference on Technology of Object-Oriented Languages and Systems*.

# Modeling and Verification of Automatic Multi-business Transactions

Min Yuan, Zhiqiu Huang, Jian Zhao, Xiang Li

*Information Science and Technology Institute,  
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China  
E-mail : { yuanmin, zqhuang, jzhao, xiangli } @nuaa.edu.cn*

## Abstract

*Web services increasingly integrate large numbers of participants to provide complex business applications. As the next step in the evolution of Web services technology, several specific protocols are being proposed to address coordinating business transactions. The specification considered here is the tentative hold protocol (THP) which facilitates the automated coordination of multi-business interactions. This work tries to enhance the reliability of multi-business by investigating formal techniques. A formal framework based on Pi-calculus for multi-business transactions is proposed. The formal specification of the THP is presented, and then verification of THP specification structural constraint among components is discussed. Subsequently, the basic model of THP is reused and shared in modeling of multi-business scenarios to enable the reasoning of multi-business scenario from the perspective of business logic requirement. The proposed framework facilitates pointing out ill-defined details in the specifications to build up a correct multi-business flow. All of the results can serve as the theoretical foundation to facilitate the coordination of multi-business interactions for implementing automatic multi-business transactions.*

## 1. Introduction

Web services are moving from their initial “describe, publish, interact” capability to a new phase in which robust business interactions are supported[4]. There are increasing needs for the enhanced transaction models that can effectively support orchestrate loosely coupled services into cohesive units of work and guarantee consistent and reliable execution. Meanwhile in a web service transactions based B2B environment, are often complex, involve multiple autonomous parties, have long duration, and may span business boundaries[10]. Just as what the BPEL specification itself states that “The notion of LRT described in BPEL is purely local and occurs within a single business process instance.[6]” It points out that this research is quite

necessary that distributed coordination regarding an agreed-upon outcome among multiple-participant services. There are a number of emerging specifications that seek to address the requirements of such web service based collaborative business transactions, such as Business Transaction Protocol (BTP), Web Services Transaction (WS-Transaction) standards etc.

However service providers usually do not allow that their resources (such as an online retailer’s items for sale) are occupied for long periods by unpredictable business operations in the loose-coupled and autonomous Web Services environment. Taking a slightly different approach to the problem, Tentative Hold Protocol (THP)[13], published as a W3C (World Wide Web Consortium) note, attempts to define a building block that can work with other technologies in order to facilitate the automated coordination of multi-business interactions as well as the creation of new opportunities to leverage the web services to improve business efficiencies.

Faced the requirements of Business-to-business (B2B) collaboration, how to design a correct business process, model and verify business transaction has become an important problem to be settled. But in fact THP does not provide a formal semantics, and there are some imprecise definitions in their informal description. It is generally accepted that formal methods are an effective approach to reducing design faults and raising trustworthiness of systems. On the other side, formal modeling of Web services transactions have become a hotspot research recently. Existing work is mainly focused on formalizing the specification of web services transactions[8] and their semantics of compensation[2] occurred within a single business process instance. Instead, here we investigate into multi-business transactions coordination among multiple-participant services.

To support the well-performing of large-scale sophisticated e-business process, we must ensure the correctness of business process and coordination. In this paper we present a formal coordination framework for multi-business transactions based on the tentative hold protocol (THP). The formal specification and model checking of multi-

business transactions are presented. The organization of the paper is as follows. Section 2 presents the overview of a formal verification framework for multi-business. Further, specification structural constraint verification of THP is presented in Section 3, and discussions of the verification of business logic requirement on THP are given in Section 4. Moreover, the related work can be found in section 5. Finally, Section 6 gives the concluding remarks and discussion of future work.

## 2. Formal Verification Framework for Multi-business Transactions

According to the IEEE Standard Glossary of Software Engineering Terminology, the correctness is defined as freedom from faults, meeting of specified requirements, and meeting of user needs and expectations. From the view of define about model checking, it is a formal verification technique that is increasingly applied to the design of industrial digital systems, and it allows to verify if the (possibly infinite) behaviors of a system satisfy a given property[3]. Verification failures result in the business specification and execution containing faults or flaws, therefore two aspects properties must be addressed. They are “specification structural constraint verification” and “business logic requirement verification” respectively. The former verifies the reliability of coordinating resources via THP, while the latter verifies whether the requirement is satisfied or not in the process of designing business process. Notice that all models of multi-business cases inherit basic processes and names from the model of THP, an overview is shown in Figure 1.

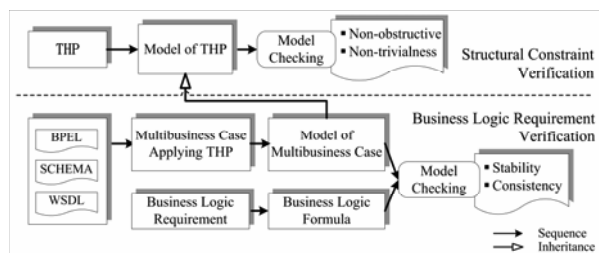


Figure 1. Formal Verification Framework

## 3. Specification Structural Constraint Verification

The major components involved in Tentative Hold Protocol[14] are displayed in Figure 2. There will be a THP coordinator on both the client and resource owner side, responsible for communicating hold requests, cancellations, etc. Also, the resource owner will provide a rules engine entity with which the resource side Tentative Hold Protocol coordinator will communicate; it shall be responsible for handling any business rule specific actions. This allows the resource owner great latitude in providing targeted

customer service with the granting of holds, specifying greater or lesser hold expirations for a given hold request, as well as the potential for notifying valued clients when some resource is being reserved by another client - allowing the preferred client the opportunity to lock in their purchase first. Tentative Hold Protocol can work with other technologies to increase their effectiveness in automating inter-business transactions. XML-based Web service standards and THP play the key role in enabling automated processes that span multiple businesses, and they facilitate the coordination of complex multi-business interactions.

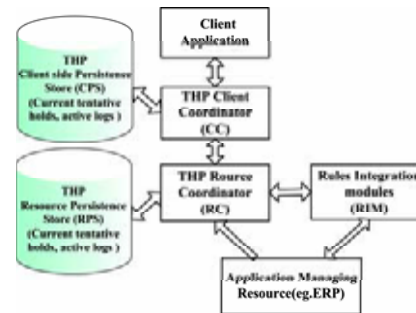


Figure 2. Components of THP

## 3.1 The Pi-calculus Model of THP

Among existing formal methods, the Pi-calculus proposed by Robin Milner[9] has drawn much attention in the field of service composition and business process modeling because of its compositionability, mobility and theoretical soundness. It describes and analyzes a concurrent mobile system via the two core concepts: processes and names. Processes interact with each other by exchanging names which are used to express the atomic interactive actions in a system. Hence, it is intuitive to adopt Pi-calculus to model the service behavior and the interaction within transactions, details about Pi-calculus refer to [9].

Processes and names are core concepts of Pi-calculus. Processes interact with each other by exchanging names, which can express the interactive actions among components of THP. Thus the establishing of the correspondences between components and processes is as follows.

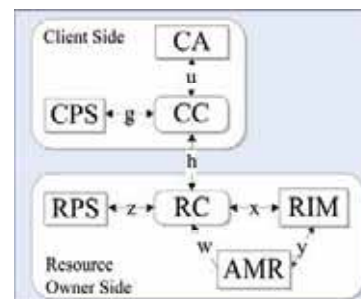


Figure 3. Flow Chart of Pi-calculus model of THP

Figure 3 shows the flow chart of THP model in Pi-calculus. Process CA, CC, CPS, RC, RPS, RIM and AMR stand for the corresponding module of THP respectively and channel u, g, h, z, x, w and y stand for channels used to transport messages among modules of THP. From the analysis above, we can build the model of THP in Pi-calculus as follows:

THP = Client() | ClientCoor() | CPS() |  
 RC() | RPS | RIM() | ARM()  
 CA : c - {Req, Resp, msg, u}  
 Client(c) = u < Req > .u(Resp).Client(c) + u(msg).Client(c)  
 CC : b - {Req, CPSResp, RCResp, stams, g, u, h}  
 ClientCoor(b) = u(Req).([Req = Archinfo]g < Req > .  
 g(CPSResp).u < CPSResp > + h < Req > .h(RCResp).  
 ([RCResp = HoldGranted]g < msg1 > .u < RCResp > +  
 u < RCResp >)) | h(stams).g < stams > .u < stams >  
 RC : d - {Req, RPSResp, RIMResp, AMRStatmsg, z, h, w, x}  
 RC(d) = h(Req).[Req = HoldRequest]x < Reqmsg > .  
 x(RIMResp).([RIMResp = Granted]  
 z < msg > .h < HoldGranted > + h < HoldDenied >).RC(d) |  
 w(AMRstatmsg).h < stams > .RC(d)  
 RIM : e - {x, y, Req, RIMReq, AMRMsg, RIMResp}  
 RIM(e) = x(Req).y < RIMReq > .y(AMRMsg).  
 x < RIMResp > .RIM(e)  
 AMR : f - {y, w, RIMReq, AMRResp, ARMStatmsg}  
 AMR(f) = y(RIMReq).y < AMRResp > .AMR(f) |  
 w < AMRStatmsg > .AMR(f)  
 RPS : g - {z, RPSResp, Req, msgRPS}  
 RPS(g) = z(Req).z < RPSResp > .RPS(g) +  
 z(msgRPS).RPS(g)  
 CPS : h - {g, Req, CPSResp}  
 CPS(h) = g(Req).g < CPSReq > .CPS(h)

### 3.2 Structural Constraint Verification among Components



Figure 4. Result of Checking

Mobility Workbench (MWB) is an automatically inference tool for Pi-calculus, which can detect whether deadlock exists in a process. It is important to ensure that the THP process is deadlock free. Figure 4 shows that all the processes in THP are deadlock free. All the participator in THP can reach the final state, and if a participator is in the ready state, the THP can move forward to the next state or return its initial state. Hence the model of THP can meet non-obstructive and non-trivialness respectively.

## 4. Business Logic Requirement Verification

THP indicates the interactive process of multi-business coordination which is implemented via the message delivery, and defines styles of various messages in accordance with THP by XML Schema. Thus we can apply THP with XML Schema defined messages to design multi-business processes. The elements and attributes of XML Schema can be mapped into Pi-calculus expressions too, and the multi-business processes can be derived from the basic THP model and these Pi-calculus expressions. Thus it can be checked to determine whether business process meets user's needs or not, which is help to omit design defects and improve the reliability of multi-business.

### 4.1 Modeling of a Scenario Applying THP

According to the XML Schema of THP, messages are made up of elements in sequence. In accordance with the grammar formulate of XML Schema, these elements should be sent according to a certain sequence. "holdHeader" is the common message header of messages in the XML Schema of THP. An scenario of applying THP, which can be seen in detail in [14] and whose sequence diagram is shown in Figure 5, is given and the scenario will be modeled using basic THP model given above, then specify some business logic requirement properties. If counterexamples are found by model checking, which means the multi-business process has some defects, solutions will be given to improve application designing.

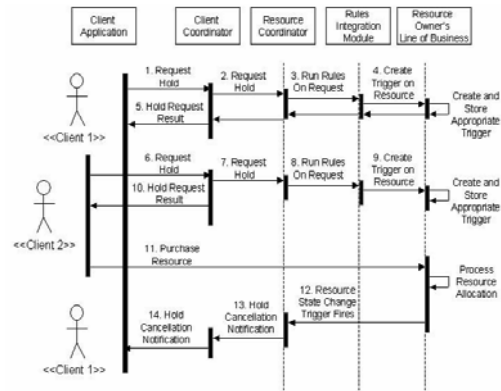


Figure 5. Sequence Diagram of a THP Scenario

CA1 and CA2 represent two clients. According to CA1's activity, CA1 can be written in Pi-calculus as below:

```
agent CA1() = CA1holdID\CA1\abc@emsoft.com < baseHeader* >.CA1holdID\CA1\abc@emsoft.com( baseHeader*; holdDuration* )|CA1holdID\CA1\abc@emsoft.com( customerHold*; cancellationReason )
```

CA2's activity includes sending request, receiving hold granted and consuming the resources, thus CA2 can be modeled by Pi-calculus as:

```
agent CA2() = CA2holdID\CA2\xyz@emsoft.com < baseHeader* >.CA2holdID\CA2\xyz@emsoft.com( baseHeader*; holdDuration* )|CA2holdID\CA2\xyz@emsoft.com < purchased >
```

Let CC1 represent the CA1's Client Coordinator, CC1 be in charge of forwarding message between CA1 and the Resource Coordinator RC, CC1's Pi-calculus description is:

```
agent CC1 = CA1holdID\CA1\abc@emsoft.com( baseHeader* ).CC1toRC < baseHeader > | CC1toRC( baseHeader*; holdDuration* ).CA1holdID\CA1\abc@emsoft.com < baseHeader*; holdDuration* > | CC1toRC( customerHold*; cancellationReason ).CA1holdID\CA1\abc@emsoft.com < customerHold*; cancellationReason >
```

CC2 denotes CA2's Client Coordinator, as modeling CC1 above, CC2's Pi-calculus model is:

```
agent CC2 = CA2holdID\CA2\xyz@emsoft.com( baseHeader* ).CC2toRC < baseHeader > | CC2toRC( baseHeader*; holdDuration* ).CA2holdID\CA2\xyz@emsoft.com < baseHeader*; holdDuration* >
```

RC resides in the resource owner, which receives request from CC1 and CC2 and then transmits responds to CC1 and CC2 after contacting with Rules Integration Module (RIM). RC can be written in Pi-calculus as follows:

```
agent RC = CA1toRC( baseHeader* ).RCtoRIM < baseHeader* >.RCtoRIM( baseHeader*; holdDuration* ).CA1toRC < baseHeader*; holdDuration* > | ROtoRIM( customerHold*; cancellationReason* ).CA1toRC < customerHold*; cancellationReason > | CA2toRC( baseHeader* ).RCtoRIM < baseHeader* >.RCtoRIM( baseHeader*; holdDuration* ).CC2toRC < baseHeader*; holdDuration* >
```

RIM denotes the Rule Integration Module in the scenario, RIM responses the requests from CA1 and CA2,

creates triggers in Resource Owner's side to detect the change of resource's state. The Pi-calculus model of RIM is:

```
agent RIM = RCtoRIM( CA1baseHeader* ).RCtoRIM < CA1baseHeader*; CA1holdDuration* >.RIMtoRO < CA1baseHeader*; CA1holdDuration* > | RCtoRIM( CA2baseHeader* ).RCtoRIM < CA2baseHeader*; CA2holdDuration* >.RIMtoRO < CA2baseHeader*; CA2holdDuration* >
```

Resource Owner (RO) receives resource allocation messages from RIM and then triggers are created. When CA2 consumes the resource, the trigger is triggered and RO sends message to RC, thus RC can update affected CA1. RO's Pi-calculus description is:

```
agent RO = RIMtoRO( CA1baseHeader*; CA1holdDuration* ) | RIMtoRO( CA2baseHeader*; CA2holdDuration* ) | ROtoRC < customerHold*; cancellationReason* >
```

The scenario model is composed of the processes above concurrently:

```
THPInstance = (CA1 | CC1) | (CA2 | CC2) | RC | RIM | RO
```

## 4.2 Model Checking and Analyzing via SAL

Symbolic Analysis Laboratory (SAL)[5], a model checker developed by Stanford Research Institute (SRI), It is a framework for combining different tools for abstraction, program analysis, theorem proving, and model checking toward the calculation of properties (symbolic analysis) of transition systems. SAL supports Linear Temporal Logic (LTL) with quantifiers, user-defined recursive data types, user-defined functions and unbounded data type. A segment of multi-business transactions are described by SAL language in Figure 6.

```
TRANSITION [
  reserving:
    C2=request --> C2'=granted;
    []
  consuming:
    (C2=granted AND C1/=purchased) --> C2'=purchased;
    []
  invalid:
    (C2=granted AND C1=purchased) --> C2'=invalid;
]
END;
main: MODULE=
  process|
  RENAME C1 TO C2,
  C2 TO C1
  IN process;
```

Figure 6. The System Transaction Described by SAL

In the scenario, two clients hold a resource together at first, and then the resource is consumed by a client. According to the business logic, the client should be informed that the resource has been consumed and his hold is invalid to avoiding the resource consumed by the two clients. Besides, each request should have its corresponding response, and these business logic requirements are related to time, thus they can be described by Linear Temporal Logic (LTL) as follows:

**Property 1:** Every request should have its corresponding response wherever in asynchronous or synchronous. The property can be written as:

$$G(C1=request \Rightarrow F(C1=denied \vee C1=granted))$$

**Property 2:** Each resource can be consumed by one client only, which can be written as:

$$G(\text{NOT}(C1=purchased \wedge C2=purchased))$$

The above two properties are the basic requirements for the business logic of THP. Of course, other properties can be defined on the basis of the specific manners in applying THP. Then the following step, we will check the two properties by SAL Model Checker named sal-smc running in Linux. The result of checking is shown in Figure 7.

```
[root@localhost bin]# ./sal-smc thp liveness1
WARNING: The environment variable SALCONTEXT_PATH does not include the current directory. Context files in the current directory will not be considered by SAL.
proved.
WARNING: Your property is only true if it is deadlock free. You should run sal-deadlock-checker for that.
```

Figure 7. Result of Checking Property 1

The model checking result shows that if the main module is deadlock free, property 1 is true, so it's necessary to check whether the main module is deadlock free. The result of checking is shown in Figure 8.

```
[root@localhost bin]# ./sal-deadlock-checker thp min
WARNING: The environment variable SALCONTEXT_PATH does not include the current directory. Context files in the current directory will not be considered by SAL.
Total number of deadlock states: 1.0
Deadlock states:
State 1
--- System Variables (assignments) ---
C1 = purchased
C2 = purchased
-----
Only 10 of 1.0 states were displayed.
```

Figure 8. Deadlock from Checking Property 1

According to the result shown in Figure 8, there is a deadlock in the main module. C1 and C2 are both in the purchased state, which is not allowed. This phenomenon can be explained by the communication delay. In ideal situation, when C2 consumes the resource, C1 will simultaneously receive the message that the resource it holds is invalid. While due to the limitation of communication in the practicing of the business processes, absolute

synchronization does not exist, thus C1 can not have notice of the resource has been consumed by C2 as soon as C2 consumes the resource, and C1 is still allowed to consume the resource. That means it might be possible that C1 and C2 are both at the purchased state together.

As shown in Figure 9, a counter-example is found while checking property 2 by SAL model checker and a path of the counter-example is also given. According to the path of the counter-example, C1 and C2 are transitioned to the granted state from the request state, and then they are both transitioned to the purchased state at last, as a result a deadlock is occurred. THP allows multiple clients to hold the same resource temporarily. When one of the clients places an order, the remaining clients receive notifications of the unavailability of the resource. However, nothing prevents a client from placing an order for a resource immediately, at which point another client might have taken the resource. This problem is mentioned in [16] too, they proposed a new reservation protocol which avoids the more need for compensating transactions because of this matter via blocking reservations. During modeling and verification of multi-business transactions, engineers can make correct decisions at the early design stage by finding ill-defined details in the specifications. So this fact illustrates enhancing the formal verification research has important meaning to guiding practice of applying Web services.

```
--- System Variables (assignments) ---
C1 = request
C2 = request
-----
Transition Information:
(module instance at [Context: thp, line(29), column(19)])
((module instance at [Context: thp, line(23), column(4)]
 (label clreserving
  transition at [Context: thp, line(12), column(21)]))
 (module instance at [Context: thp, line(27), column(13)]
 (label clreserving
  transition at [Context: thp, line(12), column(21)])))
-----
Step 1:
--- System Variables (assignments) ---
C1 = granted
C2 = granted
-----
Transition Information:
(module instance at [Context: thp, line(29), column(19)])
((module instance at [Context: thp, line(23), column(4)]
 (label consuming
  transition at [Context: thp, line(16), column(19)]))
 (module instance at [Context: thp, line(27), column(13)]
 (label consuming
  transition at [Context: thp, line(16), column(19)])))
-----
Step 2:
--- System Variables (assignments) ---
C1 = purchased
C2 = purchased
```

Figure 9. Result of Checking Property 2

## 5. Related work

Current researches on THP mainly take into consideration how to minimize the time required for clients to successfully complete their multi-transactions depending on the value of overhold size. Limthanmaphon *et al.* [7] combine the tentative hold with compensation concepts and try to minimize the possibility of transaction compensation. Younas *et al.* [15] focus on the performance of composite

transactions and propose TCP (Tentative Commit Protocol), which reduces latency in network communication and time cost of transaction processing.

There are a number of researches that analyze and verify transaction coordination protocol with formal methods, Berger *et al.* [1] formally verify the classic two-phase commit (2PC) protocol with asynchronous Pi-calculus. Qi *et al.* [12] propose the syntax and operational semantics of Membrane Calculus, which adopts the named nested membrane structure based on Committed Join Calculus to describe Web Service transactions, and makes analysis and verification on WS-AtomicTransaction (WS-AT) and WS-BusinessActivity (WS-BA) protocols. Park *et al.* [11] present a Petri net for applying THP in conjunction with two phase commit protocol, but they only model the THP and 2PC phases, and do not check whether the model has a given detailed business logic via model checking.

## 6. Conclusion

In this paper we have presented a formal coordination framework for multi-business transactions based on the tentative hold protocol (THP). Our formal solution framework using Pi-calculus and the integration of existing formal verification techniques is proposed to address the ensuring two aspects that are the verification of specification structural constraint among components and the verification of business logic requirement on multi-business scenario. The basic model of THP is reused and shared in modeling of multi-business scenarios which inherit basic processes and names from the model of THP. In analyzing the model, we find some interesting issues. The multi-business applying THP may fail to progress in some cases, which can be seen as possible cause of abnormal termination. We point out ill-defined details in the specifications, and such problem may be revealed by formal verifications. Therefore, an immediate future work is to provide a translation procedure from such schema and WSDL representations about THP messaging to our model. We also plan to model Rules Integration Modules (RIM) which is responsible for determining whether the requested resource is available. Therefore it has more need of precise semantics. As a practical matter, we are currently developing a transactional coordination framework, which as a project the Natural Science Foundation of Hunan Province of China, is used for verifying Web services transaction on resource coordination.

## 7. Acknowledgement

This work is supported by the Natural Science Foundation of Hunan Province of China with the title "Modeling and verifying Web services transaction supported for resource coordination" and Key Scientific Research Fund of Hunan Provincial Education Department (No.08A064).

## 8. References

- [1] M. Berger, and K. Honda, "The Two-Phase Commitment Protocol in an Extended pi-Calculus," *ENTCS*, vol. 39, pp. 105-130, 2000.
- [2] L. Bocchi, C. Laneve, and G. Zavattaro, "A calculus for long-running transactions," in Proc. of the 6th FMOODS, Paris, France, 2003, pp. 124-138.
- [3] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*: MIT Press, Cambridge, MA, USA, 1999.
- [4] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana, "The next step in Web services," *Commun. ACM*, vol. 46, no. 10, pp. 29-34, 2003.
- [5] L. de Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari, "SAL 2," *Computer Aided Verification in LNCS*, pp. 496-500, 2004.
- [6] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, and Y. Goland, "Web services business process execution language," *version 2.0*, OASIS Standard, May 2007.
- [7] B. Limthanmaphon, and Y. Zhang, "Web service composition transaction management," in Proc. of the 15th ADC, Dunedin, 2004, pp. 171-179.
- [8] R. Lucchi, and M. Mazzara, "A pi-calculus based semantics for WS-BPEL," *Journal of Logic and Algebraic Programming*, vol. 70, pp. 96-118, 2007.
- [9] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes. Part I / II," *Information and Computation*, vol. 100, no. 1, pp. 1-77, 1992.
- [10] M. P. Papazoglou, "Web Services and Business Transactions," *World Wide Web*, vol. 6, no. 1, pp. 49-91, 2003.
- [11] J. Park, and K.-S. Choi, "An adaptive coordination framework for fast atomic multi-business transactions using web services," *Decis. Support Syst.*, vol. 42, no. 3, pp. 1959-1973, 2006.
- [12] Z. Qi, M. Li, C. Fu, D. Shi, and J. You, "Membrane Calculus: a formal method for Grid transactions," *Concurrency and Computation: Practice & Experience*, vol. 18, no. 14, pp. 1799-1809, 2006.
- [13] J. Roberts, and K. Srinivasan, "Tentative Hold Protocol Part 1: White Paper," *W3C Note*, 2001.
- [14] K. Srinivasan, P. G. Malu, and G. Moakley, "Automatic Multibusiness Transactions," *IEEE Internet Computing*, vol. 7, no. 3, pp. 66-73, 2003.
- [15] M. Younas, Y. Li, and C.-C. Lo, "An Efficient Transaction Commit Protocol for Composite Web Services," in Proc. of the AINA, 2006, pp. 591-596.
- [16] W. Zhao, F. Kart, L. Moser, and P. Melliar-Smith, "A Reservation-based Extended Transaction Protocol for Coordination of Web Services," *International Journal of Web Services Research*, vol. 5, no. 3, pp. 64-95, 2008.



# An Adaptive Management Framework for Service Brokers in Service-Oriented Architecture

W.T. Tsai, Tszyan Chow, Yinong Chen, Xiao Wei

Computer Science & Engineering, Arizona State University, Tempe, AZ 85287-8809, U.S.A.

Contact: wtsai@asu.edu

## Abstract

*Service broker is a critical part of service-oriented architecture. A flexible and effective service broker can greatly reduce the effort for service discovery, matching, and evaluation of services and their applications. The proposed broker system provides adaptive feedback through a data-driven multi-caching mechanism. The broker is evaluated by simulation using Google Trends data. The data shows that this proposed mechanism can adapt to the changing environment efficiently and automatically.*

## 1. Introduction

Software developed in Service-Oriented Architecture (SOA) and Web services (WS) are typically developed by three parties: service providers, service brokers, and application builders [1]. A key component of a service broker is the service registry, which contains directories of contact and access information of available services hosted by the service providers. A service broker can also offer a service *repository* that hosts services submitted by service providers. Both service registry and repository are critical for efficient service discovery and matching. In fact, a service repository is similar to the concept of *service-oriented database system*. Microsoft has introduced the Service-Oriented Database Architecture (SODA) in SQL Server 2005 to handle high transaction volumes [4]. In SODA, the database is composed of a set of interconnected services. The database system can be partitioned according to predefined service boundaries that meet SOA application requirements. According to Jennings [2], SODA can be used to orchestrate data-management workflows with stored procedure instances that activate on receiving the first of one or more messages, as well as to handle each set of related messages within a transaction.

The ebXML broker contains a service repository and provides configurable and trustworthy computing and host federated entities. The federated service broker will need an adaptive broker management mechanism for efficient management in a distributed environment. Specifically, it needs to provide these

capabilities: Entities storage; ontology-based reasoning; entity association information analysis; caching services; adaptive feedback control; configuration management; and concurrency control.

This paper proposes a service-oriented broker system, which supports adaptive feedback control on its component services, and provides a reconfigurable multi-caching mechanism to improve the performance for service discovery. Service data are organized in caches, which are similar to cache groups in relational database, except the service data are usually stored in the form of XML trees instead of tuples [3]. Services can be ranked so that only service data with high ranks will be cached. Unlike the traditional database caching methods, which often perform one cache replacement strategy only at a time, the proposed management system is designed to cache an array of classified service information into different cache categories at the same time. Such caching mechanisms can be achieved in an automated or semi-automated fashion. The automated caching strategy buffers the queries and updates data continuously, and adjusts the caching mechanism correspondingly. The semi-automated caching strategy, on the other hand, allows a broker operator and/or a service consumer to assign weights to different caching criteria. The result has an immediate effect on how service data will be cached and it is possible that some participating services in the service broker can be replaced by another new service to support the new broker functionality.

In addition, service data preloading is introduced to further improve the service discovery performance using cached information. By using relational information, relevant service data can be preloaded into the cache to enhance performance.

In the rest of this paper, section 2 gives an overview of an adaptive service broker management system. Section 3 discusses the feature design of the proposed system. Section 4 presents a case study with experiment data to illustrate the key concepts. Section 5 concludes this paper.

## 2. What is An Adaptive Service Broker?

An adaptive registry provided by service brokers can store the service information in the forms of tuples

in the relational databases; of objects in the object-oriented databases; and of XML trees in XML databases. Such information may include: Data used by services; Service ontology; Business processes in BPEL and OWL-S; Service specifications in XML or WDSL; Service implementations (in Java or C#); Service usage patterns such as most frequently used services and most frequently used data; and Other infrastructures.

### 2.1. Service-Oriented Broker Design

A service broker system can be compared with a traditional database management system (DBMS). The proposed broker adopts a schema-less approach to create a reconfigurable storage environment for the system to evolve over time. Information resides in the broker system can be in one of these forms: a set of tuples, XML trees, and a block of string text. To ensure this mixed data work in a seamless manner, data conversion (e.g., from XML to tuples or from tuples to text) can be done by utilizing style sheets of data management services within the broker system when exchanging data between different parties. As a pure

service registry contains directory information only, the comparison can be made with respect to a service broker system, knowing that a service broker can be either a registry or a broker system.

In DMBS, domain ontology information needs be loaded into database before use. Such pre-processing is not needed in the proposed broker system because the consumer can subscribe service data directly.

### 2.2. Subsystems and Optimistic Operations

Figure 1 shows the design model of the proposed system. It has two sub-systems: 1) Service Broker Management System (SBMS) and 2) Service broker with Multi-Caching mechanism (SBMC). The SBMS is responsible for controlling and managing concurrency, data, data operation, and ontology. The SBMC is responsible for handling caching mechanism. SBMS is shown on the upper, while SBMC is at the lower part of Figure 1. These two parts communicate via a traditional service broker, such as UDDI, which is still needed so that standard protocols can be used in the proposed system.

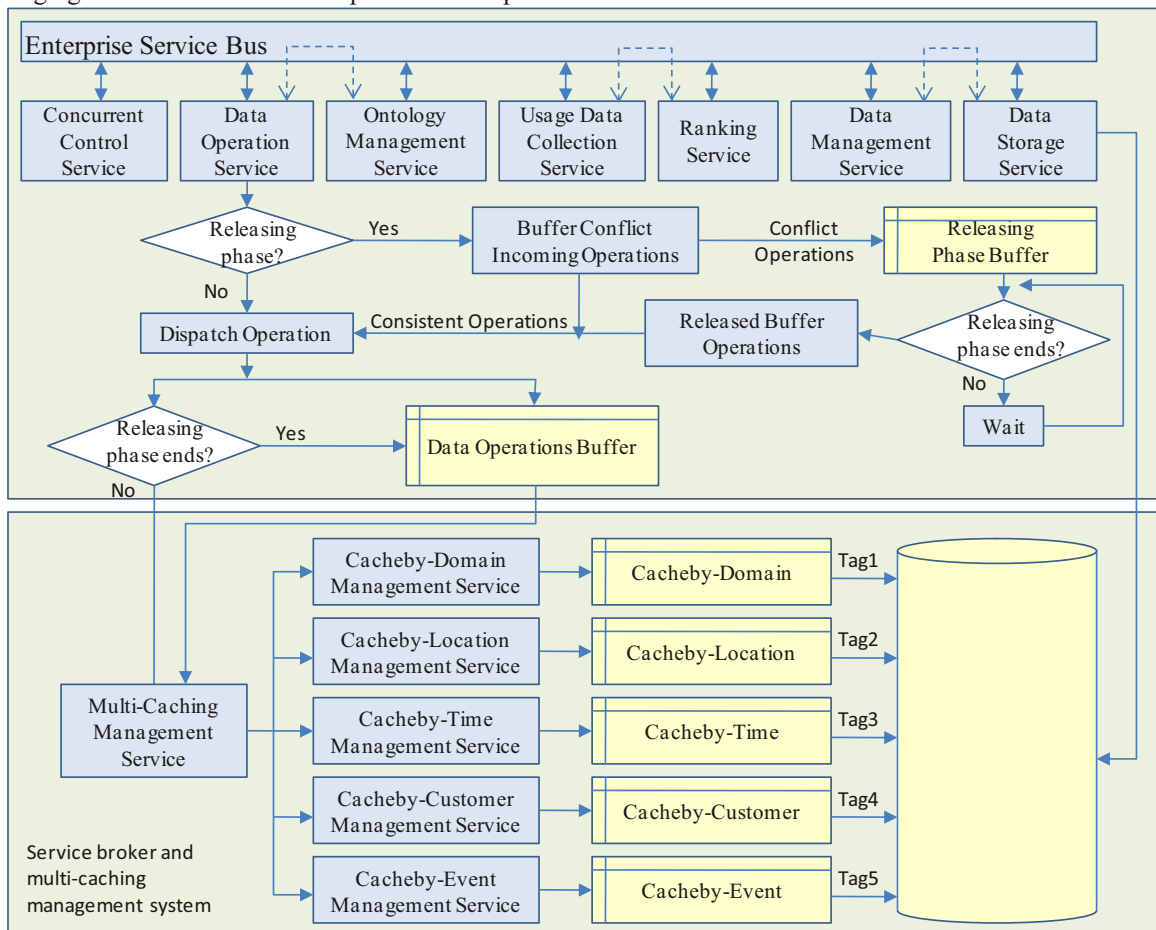


Figure 1. Adaptive Service Broker Management System

SBMS is also designed in a service-oriented manner. Specifically, it is connected to a communication bus such as EBS (Enterprise Service Bus), and its constituent services such as ontology management services and concurrency control services are connected to the bus. Thus, they can be replaced if needed. SBMS handles data in two phases:

- **Buffering Phase:** During this phase, any incoming *read-only* queries are executed in parallel, while the executions of incoming add/delete/update query operations are pended. However, if the query process involves accessing data that may change by the buffered add/delete/update operations, the query action will be set to pending as well. When the buffer is full or some other criteria are satisfied (see Section 3), the system changes to the Releasing Phase.
- **Releasing Phase:** During this phase, the pending operations in the buffer are divided into groups. While carrying out pending operations in each subgroup, other incoming service requests/ query results are continuously queued in the buffer. This non-blocking approach boosts the performance of the proposed service broker system (see Section 3). Note that when all of the pending operations are executed, the system returns to the previous phase.

SBMC handling caching mechanisms via multiple dimensions, and the details of these dimensions can be found in Section 3.2. Furthermore, this can be designed in a service-oriented manner so that the caching criteria can be dynamically changed if needed.

### 3. Adaptive Service Broker System

This paper will discuss the multi-caching approach with service ranking.

#### 3.1. Ranking Mechanism for Caching

The proposed system features service caching, along with ranking and data preloading, to shorten service response time. The caching mechanism can use the following mechanism to determine the ranking, and highly ranked services will replace lowly ranked ones.

1. Allow the users to assign weight to different ranking criteria based on the historical data or the users' preferences. For example, suppose Apple creates a service broker registry for iPod iTunes. Since the information of downloading frequency from a registered entertainment website is more interesting than the last access time. A subscriber might set the weight of the "most frequently used" ranking criteria as 90%, while set the sum of other criteria as 10%.
2. Automatically load a set of services with the highest ranks into cache. The loading operation is

controlled by the pre-specified ontology so that only the most relevant services are loaded into a specific cache. For example, a "Cache-by-Types of Songs" cache in the iTunes service registry might load services that provide rapid music downloading.

3. Whenever the historical data or user's preferences are changed, the system will dynamically update the ranking.

#### 3.2. Service Ranking

The following service ranking criteria can be used:

- **Most frequently queried:** This assumes that frequently queried services will continue to be popular in the future.
- **Most recently queried:** this assumes that a service that is recently queried has a good chance to be queried again.
- **Best quality:** This assumes that a service can be objectively evaluated using known tools.
- **Least cost:** Choose a service with least cost.
- **Most recently updated:** A recently updated service may be queried soon. For example, whenever a new video game is available, many requests will be made on this new game.
- **Most Frequently updated:** This may indicate that numerous users requested to the provider to update the service, and thus this service may be popular.

The following seven pieces of information need to be collected accordingly and let

1.  $T_{rk}$  be the time of ranking
2.  $T_{qr}$  be the time of latest query
3.  $T_{ud}$  be the time of latest update
4.  $N_q$  be the number of queries occurred in the period  $P_q$  before the time of ranking
5.  $N_u$  be the number of updates occurred in the period  $P_u$  before the time of ranking
6.  $R_{AST}$  be the rank based on the calculation with Webstar approach
7.  $R_{ce}$  be the rank based on the cost estimation.

The service ranks of the criteria above can be calculated with the following criteria:

- **Most frequently queried:** Arrange services in decreasing order of  $N_q$ , and the  $R_{mfq}$  rank of a service within  $P_q$  equals to its index in this sequence.
- **Most recently queried:** Arrange services in the increasing order of  $T_{rk} - T_{qr}$ , and the  $R_{mrq}$  rank of a service equals to its index in this sequence.
- **Best quality:** Arrange services in the increasing order of  $R_{AST}$ , and the  $R_{bq}$  rank of a service equals to its index in this sequence.

- **Least cost:** Arrange services in the increasing order of  $R_{ce}$ , and the  $R_{lc}$  rank of a service equals to its index in this sequence.
- **Most recently updated:** Arrange services in the increasing order of  $Trk - T_{ud}$ , and the  $R_{mru}$  rank of a service equals to its index in this sequence.
- **Most frequently updated:** Arrange services in decreasing order of  $N_u$ , and the  $R_{mfu}$  rank of a service within  $P_u$  equals to its index in this sequence.

The overall rank of a service can be calculated with

$$R_{overall} = \sum_{i=1}^N w_i R_i$$

Where,  $N$  is the number of ranking criteria with the default value 7,  $R_i$  is the rank according to the  $i^{\text{th}}$  criterion, and  $w_i$  is the weight of a rank criterion in terms of percentage.

### 3.3. Multi-Caching

As different consumers may have different preferences on the services registered, it is the broker's responsibility to ensure efficient service matching and allocations to the corresponding consumers. For example, when sending service inquiries to the service broker, a service consumer may only be interested in services hosted in the location closest to them. In this example, it will be desirable for the service broker to use location-based caching technique to cache these services.

Multi-caching is a way to provide efficient services by reducing multiple round trips to the server and increasing the availability of services. Using a multi-caching approach, a service broker system is equipped with multiple caches. Each cache can store service data in response to various ranking criteria and categories. For example, the ranking option may include: Domain, Location, Customer, Time period, and Special events.

For a given service registry, the information stored in the broker will be searched by the users from different locations during different time periods. If the information is cached according to their domains, it helps the users to retrieve the related services rapidly. For example, assume that there exists a service registry for sports-related services that caches domains such as American sports leagues NBA, NFL, and MLB. A service consumer requesting NBA related services can easily query the services reside in the NBA domain service cache. If the NBA services are also cached by locations, the service consumers in Phoenix area may query the services for the Suns and service consumers in Chicago may query the services for the Bulls easily. During NBA postseason games, services related to the teams in the postseason will be cached. When it is time for NBA all stars weekend,

the services such as all stars game tickets selling services, all stars weekend hotel reservation services, and travel planning services related to the host city will be cached for convenient access.

### 3.4. Service Data Preloading

The performance of service discovery can be further enhanced by automated preloading of related services when adding a new service into the cache. In the registry management system, *relational preloading* is used as a preloading service, which loads services related to the newly cached service based on the relationship specified in the service specification, instead of other factors such as locality.

The relationships among services are specified in the service specification such as PSML (Process Specification & Modeling Language) [5] and/or in the ontology model. The broker management system analyzes a variety of ontology relationships to support relational preloading, as listed in the Table 1.

**Table 1.** Ontology Relationships Definition

Relationships	Definitions
<i>BelongTo</i>	The service belongs to a specific domain.
<i>RelateTo</i>	A domain is related to another domain.
<i>LocateAt</i>	The service is located at a specific location (city/state/country).
<i>OccurDuring</i>	The service's contents focus on activities during a specific time period.
<i>InvolveWith</i>	The service is involved with other services.
<i>CloseTo</i>	An owner/customer of a service has close relationship to other person.
<i>FocusOn</i>	The service provides supports for a particular event.

Each service may include specifications that cover multiple of the above relationships. When a service is added into the cache, the highly ranked services with same ontology relationships may be swapped into the multi-cache system. For example, when a service consumer uses the browser located in Phoenix to search for the best quality NBA services through a service broker, the sport services that with the highest rank in reporting NBA news will be swapped into Cache-By-Domain. Similarly, any highly ranked web services that are related to Phoenix Suns will also be loaded into Cache-By-Location.

### 3.5. Data Operation and Optimization

This section presents an optimization approach to data operation service called *buffer-and-release*. The key idea is to buffer the incoming add/delete/update operations as well as partial query operations in the Data Operations Buffer and allow the rest of the query operations run concurrently until certain criteria are satisfied. When the criteria are met, the system will release and execute all the holding operations until the Data Operations Buffer is empty. A query operation is buffered if and only if it tries to

access the data that will be modified by the existing data operations in the Data Operations Buffer.

The Data Operation Service has two phases: the buffering phase and the releasing phase. During the buffering phase, the Data Operation Service accepts all the incoming query operations, buffers add/delete operations and partial query results until certain criteria are satisfied. When the system is *ready*, it switches to the releasing phase which will perform the add/delete/query operations stored in the data operations buffer. Once the Data Operations Buffer is empty, the state is changed back to the buffering phase. Let

1.  $N_a$  be the number of add operations,
2.  $N_d$  be the number of delete operations
3.  $N_q$  be the number of query operations
4.  $N_{esr}$  be the number of data entities in the service broker
5.  $N_{eom}$  be the number of data entities involve with the add/delete operations in the Data Operations Buffer
6.  $P_{bf}$  be the pre-specified policies to regulate the behaviors of the data operations

The proposed system supports the following buffer releasing criteria:

1.  $N_a + N_d > N_t?$ , where  $N_t$  is the pre-specified threshold value.
2.  $\frac{N_a + N_d}{N_q} > P_{tpq}?$ , where  $P_{tpq}$  is the threshold percentage with respect to  $N_q$
3.  $N_{eom} > N_{te}?$ , where  $N_{te}$  is the maximum number of data entities that are allowed to be involved with the operations in the buffer
4.  $\frac{N_{eom}}{N_{esr}} > P_{tesr}?$ , where  $P_{tesr}$  is the maximum percentage between  $N_{eom}$  and  $N_{esr}$
5.  $P_{bf}$  is violated? For example, a policy might specify the maximum number of add operations is 100 times a day.

The above criteria can be dynamically selected and replaced in the runtime based on the analysis results after they are implemented as services.

The execution can be optimized by dividing data operations buffer into parts and only one segment will be executed at a time. When releasing operations from a segment, operations that involve accessing the same data in current partition (*conflicted incoming operations*) will be buffered / pended, and other non-conflicted incoming operations (*consistent incoming operations*) will be dispatched and executed immediately. The pending operations will also be dispatched and executed after the releasing phase ends. This non-blocking parallel design

amplifies the performance of the overall data operation execution process.

There are multitudes of ways to partition database schema. The partitioning approach that we adopt is XML-file-based, for the most common format to store the data in the service broker are XML files. With operation partition, the execution process during the releasing phase follows the algorithm below.

```

Divide the data operations in the Data Operations Buffer
into N parts;
for (int i = 0; i < N; i++){
    Pick up the ith part;
    While (not all the operations in the ith part are
executed){
        Pick up the next operation from the ith part which
does not access the same data as the operations
currently being executed;
        Start executing this operation;
        if (no such operations are left){
            while (the load is not full){
                Execute incoming query operations that
do not access the same data as the
operations currently being executed;
            }
        }
    }

```

## 4. Experiments

To fully understand how the adaptive feature of the service broker system works, it is important to recognize how the system behavior changes in accordance with the cached services and the users query patterns. When the user first interacts with the broker, the performance gain from the cache is zero because no cache information is available. Once the initial search is performed, all subsequent searches may require less time to perform because. Figure 2 shows the generalized cache cycle in the proposed system using the service access and cache data collected during the events of the NFL Super Bowl 2007 and NBA All Start 2007. The y-axis on the left represents the number of services retrieved from the cache and the one of the right represents the percentage of the cached services that are relevant to the new query. The x-axis indicates the time when the search happens.

In Figure 2, the two distinctive bell curves, S1 and S2, are generated to represent two different domains that are frequently queried by the users in a specific time period. Both S1 and S2 first started from zero cache hit and then gradually increase the hit rate as services are being cached after each look up. The service retrieval time is optimal when it achieves 90% cache hit. The reason being is that the service subscribers are retrieving 90% of the desirable services directly from the cache instead of the remote servers. However, due to the dynamic nature of the system, as a new service query arrives, the presently cached items will be gradually replaced by the next

popular services. As a result, the cache hit % for the previously popular service will be lower, and S1 and S2 overlap with each other.

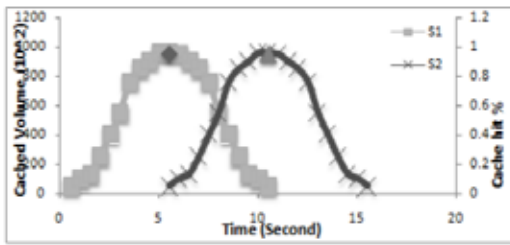


Figure 2. Sample Service Retrieval Cycle

To test the effectiveness of the proposed service caching and preloading techniques discussed in the preceding sections, a stimulation analysis has been performed to compare different service calls along with the cache information but uses actual data from Google Trends [10]. The stimulation program is written in C#. Google Trends analyze how frequently a subject has been searched on Google over time and across the global. Data obtained from Trends are normalized and scaled relative to the average traffic of the search item. To stimulate the service broker system, the actual data for NFL and NBA search in the United States during the February 2007 has been collected from Trends.

The Search Index Volume (SVI) on the y-axis indicates the popularity of the search item. Presumably, all items start at 0.0 SVI. As the search of a particular item become more frequent, the SVI will be increased with respect to its norm. For instance, in Figure 3 a noticeable spike (1.65 SVI) between February 2<sup>nd</sup> and February 6<sup>th</sup> indicates the search traffic is 1.65 times higher than the usual for the NFL. In this experiment, only the data from February 1<sup>st</sup> to 22<sup>nd</sup> 2007 is used. The SVI value has been augmented by 100 to represent the virtual data volume and the scale of the search data is also compacted from daily to millisecond for the stimulation program. All service query activities are artificially generated according to the search pattern observed in Table 2. The transformed version of Trends graph is shown in Figure 4.

Table 2. Google Trends Data for NFL and NBA in February 2007

Day(2007)	2/1	2/2	2/3	2/4	2/5	2/6	2/7	2/8	2/9	2/10	2/11
nfl	0.55	0.56	0.6	0.58	0.44	0.51	0.52	0.55	0.56	0.62	0.71
nba	0.74	0.85	0.98	1.68	1.64	0.81	0.68	0.66	0.65	0.69	0.81
	2/12	2/13	2/14	2/15	2/16	2/17	2/18	2/19	2/20	2/21	2/22
	0.51	0.5	0.53	0.6	0.73	1.03	1.31	1.11	0.64	0.66	0.86
	0.61	0.61	0.55	0.58	0.6	0.58	0.55	0.55	0.55	0.56	0.59

## 5. Summary

This paper presents an adaptive service-oriented registry management framework for service registries. This framework supports adaptive feedback control, reconfigurable service ranking, caching and

preloading. The data operations are optimized to improve the performance of the execution process. A case study and experiment data are presented in this paper to demonstrate the techniques in the framework. A stimulation analysis using Google Trends data is also performed to demonstrate the effectiveness of the proposed framework for service discovery and matching for the service consumers.

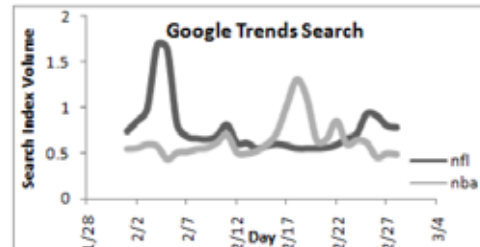


Figure 3. Analysis using Google Trends Graph

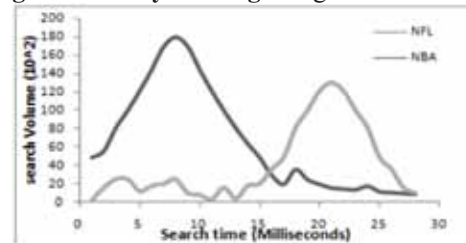


Figure 4. Google Trends Reproduction

## References

- [1] Y. Chen, W.T. Tsai, Distributed Service-Oriented Software Development, Kendall/Hunt Publishing, 2008.
- [2] R. Jennings, "Program SQL Server 2005' Service Broker", June 12<sup>th</sup>, 2006, available at <http://www.ftponline.com/vsm/2006%5F06/magazine/features/rjennings/>.
- [3] Y. Kim, S. H. Park, T. S. Kim, J. H. Lee, and T. S. Park, "An Efficient Index Scheme for XML Databases", SOFSEM 2006, pp. 370-378.
- [4] Microsoft, "Why Consider a Service-Oriented Database Architecture for Scalability and Availability", White Paper, November 2005.
- [5] W. T. Tsai, X. Wei, Z. Cao, R Paul, Y. Chen, and J. Xu, "Process Specification and Modeling Language for Service-Oriented Software Development", 11th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS), Sedona, AZ, March 2007, pp.181-188.
- [6] H. Jiang, C. Ho, L. Popa, W. Han, "Mapping-Driven XML Transformation", In Proceeding 16<sup>th</sup> International World Wide Web Conference (WWW), Banff, Canada, May 2007.
- [7] IBM WebSphere software, <http://www-306.ibm.com/software/websphere>
- [8] IBM Autonomic computing project, <http://www-03.ibm.com/autonomic/>
- [9] SmartChannels Repair, BBN, 2002, [http://aai.bbn.com/cougaar/Smart Channels SelfHealing\\_final.pdf](http://aai.bbn.com/cougaar/Smart Channels SelfHealing_final.pdf)
- [10] Google Trends, <http://www.google.com/trends>

# Requirements Discovery Based on RGPS Using Evolutionary Algorithm

Tao Peng, Bing Li, Weifeng Pan, Zaiwen Feng  
State Key Lab of Software Engineering, Wuhan University, 430072, Wuhan, China  
School of Computer, Wuhan University, 430072, Wuhan, China  
[tao.peng@163.com](mailto:tao.peng@163.com) [libing@sklse.org](mailto:libing@sklse.org)

## Abstract

*Requirements discovery is the base of software engineering, and it has a great effect on the software development. The requirements are usually elicited through the communication with users in practice, but this method is not suit for networked software. In this paper, we present a novel method for requirements discovery in networked software. It represents the initial requirement with tree based on RGPS (Role-Goal-Process-Service) and uses evolutionary algorithm to find other requirements similar to the given initial one. The experiment results show that the method proposed in this paper not only can find out the requirements similar to the initial one, but has better adaptability and higher efficiency than the traditional nested loop method used in the case of complex requirements and domain ontologies. It is promising.*

## 1. Introduction

Networked Software is a novel software paradigm which is proposed by our research group<sup>[1]</sup>. The Networked Software is much different from the traditional software, which elicits requirements from users online, analyzes users' requirements statements, creates users' requirements model, selects model assets according to users' requirements model, discovers and composes web services to create a system to satisfy users in an acceptable period of time.

The traditional method for requirements discovery is interacting between developer and users, and the researches of requirements discovery have been carried out from different phase of software developing<sup>[2]</sup>, or from special software system<sup>[3]</sup>. However, for the characteristics of networked software, we need a new method for requirements discovery.

On networked environments, there is a lack of interaction between users and developers. Users don't know what software they need, and developers don't know who will use their software. RGPS<sup>[4]</sup> is a requirements meta-modeling framework for networked software which starts from analyzing requirements and ends with providing solutions based on services. RGPS is a bridge between users and developers, and makes them work separately well. Generally, there is no service can fulfill user's requirement directly, so we need a method to mining user's potential requirements and provide personalized services to them. All of these should be completed online, so the similarity of requirements and time consuming have high priority.

Based on RGPS, the Nested Loop (NL)<sup>[5]</sup> method can be used to discover the requirements. It first constructs the warehouses of similar words for each element elicited from user's requirement. Then it uses nested loop to assemble the words in each warehouse, and keeps the assembled requirements which have fitness bigger than a given threshold. The advantages of NL are that it can be easily implemented, and it can discover all required requirements, but its efficiency is very poor especially in the case of complex requirements and domain ontologies.

For the problem proposed above. This paper presents a new requirements discovery method named RERD. It represents user's requirement with tree, and adopts evolutionary algorithm (EA)<sup>[6]</sup> to search requirements similar to user's requirement. The numerical experiment shows that compared with the NL method, the RERD method has better adaptability and higher efficiency than NL in the case of complex requirements and domain ontologies.

The reminder of the paper is organized as follows: section 2 gives a brief introduction RGPS and EA. Section 3 details the RERD method. Section 4 is the numerical experiment. And we conclude this paper in section 5.

## 2. RGPS and EA

As we talked above, RERD method is based on RGPS and EA, so in this section we will give a brief introduction to RGPS and EA at first.

### 2.1. RGPS

RGPS<sup>[4]</sup> is a service-oriented requirements meta-modeling framework for networked software. It can be used to guide requirements modeling in networked software. According to the characteristics of requirements in networked software, the RGPS framework consists of four layers: Role, Goal, Process, and Service. Web Ontology Language (OWL) is adopted as concrete syntax to describe Role Layer and Goal Layer, whereas Process Layer and Service Layer are described by OWL-S.

Role Layer is used to depict the actors, roles played by actors and context of the actors. A role can take charge of several goals.

Goal Layer depicts goals and properties of goals. Goals can be classified into Functional Goals and Nonfunctional Goals. A Functional Goal has an Operation, an Object and a Manner. Functional Goals can be decomposed into several sub-goals. The decomposition will go on until all the sub-goals are Operational Goals which can be achieved by Processes. Besides Functional Goals, Nonfunctional Goals are also defined in the Goal Layer. Nonfunctional Goals are classified into Quantitative Nonfunctional Goals and Qualitative Nonfunctional Goals.

Because Process Layer and Services Layer have little relationship with the work in this paper, we don't introduce it. For more details, please refer to Ref. 4.

### 2.2. EA

EA (Evolutionary algorithm)<sup>[6]</sup> uses the mechanisms inspired by biological evolution (reproduction, mutation, recombination, and selection) to guide the learning and searching direction. EA can find the best individual in a population with evolving. However, in our method, we already have the best individual which is user's requirement, what we need to do is to find the population of similar requirements. In order to use EA to discover the requirements, we should modify the traditional EA in many aspects such as individual representation, the recombination operation, the mutation operation, and fitness function

## 3. Construction of RERD

Figure 1 gives the overview of the RERD method:

Step1: Model the requirements with RGPS, and extract the elements correspondingly  
 Step2: Construct the Mutation Pool for each element in the initial requirement  
 Step3: Represent the requirements with a tree structure, namely requirement tree  
 Step4: Evolve the requirement tree with EA, and return a population of similar requirements.

**Figure 1 The steps of RERD**

### 3.1. Element Extraction

To represent user's requirement, we should first elicit the elements such as behavior, condition, event, reaction, constraint status, etc from the requirement<sup>[7]</sup>. Because this paper only focuses on the discovery of potential user requirements, only Role layers and Goal layer will be used. Suppose that one requirement is described as "grandpa wants to travel by car", after eliciting, we get four elements: Role, Goal, Purpose and Manner. The Goal element is an abstract element without a value, but containing the Purpose and Manner elements. Where "Old\_Man" is the value (or word) of Role, "Travel" is the value of Purpose, and "By\_Car" is the value of Manner.

### 3.2. Mutation Pool

Mutation pool is a warehouse of words similar in semantic. When the mutation operation is selected, we will choose a word from the mutation pool to replace the corresponding word of the parent requirement. To prevent the mutation operation produce too much useless requirement, we only construct the mutation pool for all non-abstract elements of each initial requirement. The child requirements will inherit the mutations pools from parent requirements, no matter the operation is recombination or mutation. We can construct the mutation pool by following steps:

- 1) Add the word (value) of the non-abstract element into warehouse;
- 2) Add the words of domain ontologies with similarity value bigger than threshold  $Sim_{lexical}$  into the warehouse.  $Sim_{lexical}$  is a parameter can be used to change the size of mutation pool.

The function used to evaluate similarity between any pair of words' is written as:<sup>[8]</sup>

$$Sim_{lexical}(O_1, O_2) = \frac{\alpha \times (l_1 + l_2)}{(Dis(O_1, O_2) + \alpha) \times \max(|l_1 - l_2|, 1)}$$

Where  $Sim_{lexical}(O_1, O_2)$  is the similarity of word  $O_1$  and word  $O_2$ ,  $Dis(O_1, O_2)$  is the distance between



two words;  $l_1$  is the level of word  $O_1$ ;  $l_2$  is the level of word  $O_2$ ;  $\alpha$  is an adjustable parameter bigger than 0. For more details, please refer to Ref. 8.

We should construct three types of mutation pool for the requirements in section 3.1. They are Role mutation pool, Purpose mutation pool, and Manner mutation pool.

### 3.3. EA in RERD

In this section we will detail the EA that used in RERD in the following aspects: individual representation, recombination operation, mutation operation and the fitness function.

**3.3.1. Individual Representation.** The individual representation of GP (Genetic Programming) [9], the tree structure, will be introduced into our method to represent the requirements, which can be constructed by following three steps:

- 1) The root node of the tree is “Requirement”;
- 2) The intermediate nodes of the tree are the abstract element, for example, the Goal element;
- 3) The leaf nodes of the tree are the elements which have a value (word), for example, the role element.

Figure2 gives an illustration of the representations of the requirement “grandpa wants to travel by car”.

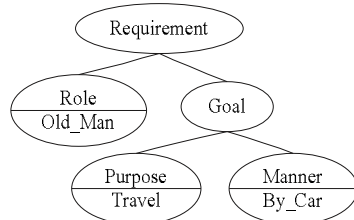


Figure 2 Requirement tree

**3.3.2. Recombination Operation.** Recombination is an evolutionary operation to produce new individuals. The new individual produced by combining the different parts of parent individuals. In RERD, the recombination operation is implemented by following steps:

- 1) Choose two parent requirements randomly,
- 2) Choose a recombination point randomly, and exchange the subtree rooted in the recombination point to produce two new child requirements.

The recombination points of the two parent requirements should be at the same point. Figure 3 gives an illustration of the recombination operation: one parent requirement is “grandpa wants to travel by car”, and the other parent requirement is “Harry goes to school by bike”, and the recombination point is

“Manner” node. After recombination, we get two child requirements that are “grandpa wants to travel by bike” and “Harry goes to school by car”.

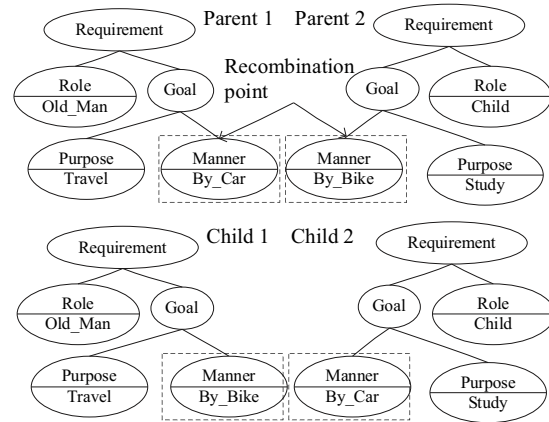


Figure 3 Recombination operation

**3.3.3. Mutation Operation.** The mutation operation is beneficial to the diversity of individuals. In RERD, it only operates one parent requirement to generate new individual by changing some specific genes randomly which can be implemented by the following steps:

- 1) Choose a mutation point in the requirement tree, which is either the intermediate node or the leaf node;
- 2) If the mutation point is at a leaf node, randomly choose a new word from the corresponding mutation pool, and replace the chosen old word. If the mutation point is at an intermediate node, do mutation operation for every leaf node of the subtree rooted in this node.

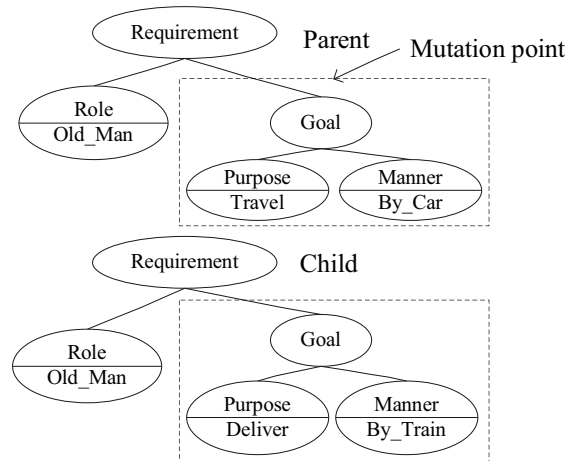


Figure 4 Mutation operation

Figure 4 illustrates the mutation operation: the parent requirement is “grandpa wants to travel by bike”. The mutation point is “Goal”. After mutation,

we get the child requirement, “grandpa delivers goods by train”.

**3.3.4. Fitness Function.** After recombination and mutation operation, we should evaluate the fitness of the child requirements to discard the child requirements with fitness smaller than a given threshold. The fitness value is used to evaluate the similarity between the child requirement and the initial requirement. The fitness function used in RERD can be written as:

$$Sim_{req} = \sum_{k=1}^M \alpha_k Sim_{lexical}(O_k', O_k)$$

Where  $M$  is the number of leaf nodes.  $O_k$  is the word of the leaf node in initial requirements, and  $O_k'$  is the word of the corresponding node in child requirements.  $Sim_{lexical}(O_k', O_k)$  is the similarity between  $O_k$  and  $O_k'$ .  $\alpha_k$  is the weight bigger than 0 on the node.

**3.3.5. Terminate condition.** The terminate condition in EAs usually has two types: one is controlling the maximum evolving generation (MEG); the other one is judging whether the requirement population has been unchangeable in a certain number of generations. To control the time the RERD method may cost, the first type of terminal condition will be chosen here.

## 4. Experiment Analysis

In this section RERD will be applied to two practical requirements discovery cases. The results of the experiments show that the proposed method can get similar requirements as the traditional methods, with a better time efficiency. The experiments are implemented with Java on Eclipse-europa platform in a Dell notebook with CPU Intel T7250, Memory 1G, and OS Windows XP.

### 4.1. Numerical Experiment

Suppose there is an initial requirement: “someone will watch Olympic Games by train”. The elements extracted are shown in table 1. The three domain ontologies are available at [10]. The parameter settings are shown in table 2:

**Table 1 Elements extracted**

	Role	Purpose	Manner
Value	Man	Enjoy Olympic Game	By Train

**Table 2. Parameter settings**

$Sim_{lexical}'$	$\alpha_{role}$	$\alpha_{purpose}$	$\alpha_{manner}$	$Sim_{req}'$	MEG
1	1	1	1	10	60

The program finds 18 similar requirements within 260ms. The 18 similar requirements shown in table 3 are ranked according to  $Sim_{req}$  from small to large.

In the following, we will take the 1<sup>st</sup> and 18<sup>th</sup> requirement as examples to analyze why they are chosen by RERD. Since the 1<sup>st</sup> requirement’s Manner is different from the initial requirement, we should calculate the fitness of the requirement. According to the formulas and the relationship of elements in ontologies, the 1<sup>st</sup> requirement will be kept in the population for its fitness is 13.2 larger than the threshold  $Sim_{req}'$ . By the same token, the 18<sup>th</sup> requirement will also be kept for its fitness is 14.3. The fitness value of the 18<sup>th</sup> requirement is bigger than the 1<sup>st</sup> requirement, so the 18<sup>th</sup> requirement is more similar to the initial requirement than the 1<sup>st</sup> one.

**Table 3 Similar requirements**

No	Role	Purpose	Manner	$Sim_{req}$
1	Man	Enjoy Olympic Game	By Plane	13.2
2	Old Man	Enjoy Olympic Game	By Train	
3	Father	Enjoy Olympic Game	By Train	
4	Man	Watch Basketball Game	By Train	
5	Child	Enjoy Olympic Game	By Train	
6	Man	Watch Football Game	By Train	
7	Woman	Enjoy Olympic Game	By Train	
8	Old Woman	Enjoy Olympic Game	By Train	
9	Baby	Enjoy Olympic Game	By Train	
10	Man	Enjoy Olympic Game	By Coach	
11	Middle Couple	Enjoy Olympic Game	By Train	
12	Man	Enjoy Olympic Game	By Ship	13.7
13	Man	Enjoy Olympic Game	Long Travel Manner	
14	Man	WatchSportGames	By Train	
15	Person	Enjoy Olympic Game	By Train	
16	Man	Watch Swimming	By Train	14.3
17	Man	EnjoyJalor	By Train	
18	Man	EnjoyHorsemanship	By Train	

### 4.2. Data Comparison

In this section the proposed RERD will be executed 10 times under different size of mutation pool, and compared with the traditional NL method in two aspects: the requirement searching capacity (the number of requirements obtained) and the time efficiency. And the data are averaged. In RERD, the size of mutation pool can be adjusted by setting different similarity threshold  $Sim_{lexical}'$ . Here the  $Sim_{lexical}'$  will be set to be the following values: 1, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, and 0.5.

The comparisons between RERD and NL in requirement searching capacity are shown in figure 5. On the whole prospective, the number of requirements found increases with  $Sim_{lexical}'$  decreasing. When  $Sim_{lexical}'$  changes from 1 to 0.7, the size of mutation pool is small, and the evolving generation is big enough to find all requirements, so the number of

requirements found by RERD is same as that of NL. However, with the  $Sim_{lexical}$  decreasing, the size of mutation pool increases, making the RERD can't find all the requirements only with 2 or 3 requirements being missed compared with NL.

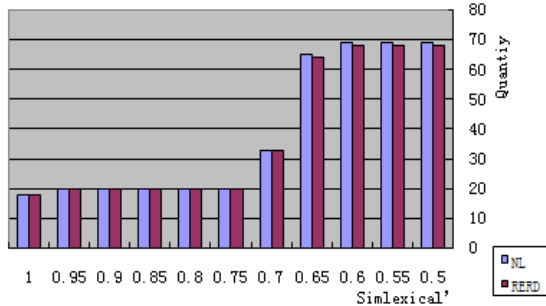


Figure 5 Comparison of requirements number

The comparisons between RERD and NL in time efficiency are shown in figure 6. From the figure, we can find that the consuming time increases with the  $Sim_{lexical}$  decreasing. When  $Sim_{lexical}$  changes in the range from 1 to 0.7, the consuming time of RERD is larger than that of NL by roughly 80ms. When  $Sim_{lexical}$  change in the range from 0.65 to 0, the consuming time of NL increases at a high speed, near to 2s. Conversely, the consuming time of RERD increases slowly, still less than 1s.

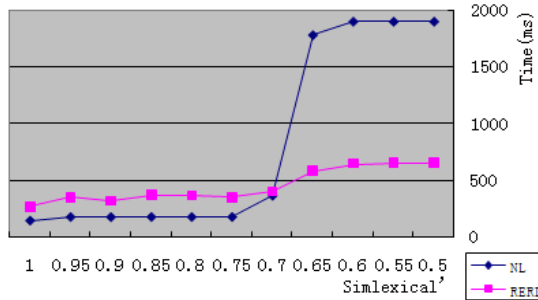


Figure 6 Comparison of consuming time

Though NL can be used to find all the requirements, it is very time consuming in the case of complex requirements and domain ontologies. However, EA with the characteristics of self-adaptation, makes RERD can find requirements more effectively. It is acceptable that the RERD to save the time at the cost of missing several requirements.

## 5. Conclusion

A method for requirements discovery based on RGPS and evolutionary algorithm is proposed in the paper. And the construction of evolutionary algorithm for requirements discovery is also proposed. It is

proved experimentally that the RERD method can find requirements similar to the user's requirement. Furthermore, compared with the NL method, the RERD method can be easier to control the time consuming by setting maximum evolving generation and more effective with acceptable missing few requirements in the case of complex requirements and domain ontologies. So the RERD method is more suitable for the online processing course of networked software.

## Acknowledgements

This work is supported by the National Basic Research Program of China (973) under Grant No.2007CB310801, the National High Technology Research and Development Program of China (863) under Grant No.2006AA04Z156, the National Natural Science Foundation of China under Grant No.60873083, 60703018, 60703009 and 60803025.

## 6. References

- [1] He Keqing, Liang Peng, et al. Requirement emergence computation of networked software [J]. Frontier of Computer Science in China, 2007, 1(3): 322-328
- [2] RR Lutz, IC Mikulski. Resolving Requirements Discovery in Testing and Operations [C], the 11th IEEE International Requirements Engineering Conference, 2003.
- [3] N Seyff, F Graf, et al. Mobile Discovery of Requirements for Context-Aware Systems [C], REFSQ 2008:183-197.
- [4] Wang Jian, He Keqing, Li Bing, et al. Meta-models of Domain Modeling Framework for Networked Software [C], In Proc. of the Sixth GCC Conference, Urumchi, China, July 2007.
- [5] [http://en.wikipedia.org/wiki/Nested\\_loop\\_join](http://en.wikipedia.org/wiki/Nested_loop_join)
- [6] Liu Yong, Kang Lishan, et al. Genetic Algorithm [M]. Beijing: Science Press, 1997.
- [7] Liu Wei, Peng Rong, et al. Heavyweight Semantic Inducement for Requirement Elicitation and Analysis [C], the 3rd SKG, pp: 206-211, Oct. 2007
- [8] Wu Jian, Wu Zhaohui, et al. Web Service Discovery Based on Ontology and Similarity of Words [J]. Chinese Journal of Computers, 2005, 28(4): 413 - 417.
- [9] Zhang Zong-hua, Zhao Lin, et al. An Introduction and Survey of Application of Genetic Programming [J]. Computer Engineering and Applications, 2003, 39(13): 94-97.
- [10] <http://61.183.121.131/973/ontology.rar>

# Mediation Based Variability Modeling for Service Oriented Software Product Lines

Mohammad Abu-Matar  
George Mason University  
Fairfax, VA USA  
mabumata@gmu.edu

*Abstract* – Service Oriented Architecture (SOA) has emerged as a model for distributed software development that promotes agility and large scale reuse. Software product lines (SPL) promote agile and flexible application development for software families. SPL development relies on feature models to describe the configurations of software member applications. To develop service-oriented SPL, variability analysis must be applied to existing services that are decoupled from service requesters. In this paper, we rely on SPL engineering principles to develop a service oriented variability mechanism based on the Enterprise Service Bus (ESB) concept. We start with service orchestration models provided by domain business analysts. Then, we develop feature models similar to those of SPL to model commonality and variability requirements. Then, we develop Mediation modules to model the variability requirements of service oriented applications. Mediation modules reside in the ESB layer which is above the services layer. This way, we model variability independent of the services layer and yet satisfy variable requirements of multiple service oriented applications. We then show how variant member applications could be composed and executed using the ESB in any SOA environment.

## 1. Introduction

Software Product Lines (SPL) are families of software systems that share common functionality, where each member has variable functionality [1]. The main goal of SPL is the agile and rapid development of member systems by using reusable assets from all phases of the development life cycle. This goal is similar to the goal of Service Oriented Architecture (SOA) where agile and flexible application development is a common theme.

An essential requirements modeling phase in Software Product Lines Engineering (SPLE) is Commonality and Variability Analysis (CVA) where the common and varying features of SPL member applications are outlined. CVA is commonly expressed in Feature Models based on the SPL common, optional, and alternative use cases.

In SPL, a variable component architecture is specified up front based on the feature model. In SOA, the main elements of the architecture are services normally provided by outside sources. A major characteristic of SOA is the decoupling between the business process layer and the service layer [2] where both layers can evolve independently of each other. Therefore, services in the service layer have no notion of the process layer or any

other clients. More importantly, services in the service layer have no notion of any clients' variability concerns.

Variably modeling for service oriented applications should be performed independent of the service layer, since normally client applications have no control over the provided services that can be located and consumed from anywhere over the Web. Hence, the challenge is *how to design variability concerns of service oriented applications independent of the service layer and yet use the services in a way that satisfies multiple application scenarios.*

In this paper, we use SPL variability modeling principles to model variability of service oriented applications using the Enterprise Service Bus (ESB) [5] layer.

An ESB is an intermediary layer present in most SOA environments that decouples service providers from requesters. The ESB is normally built off a Message Oriented Middleware MOM [5]. Clients and services *plug* into the ESB seamlessly without worrying about communication or infrastructure details. In addition, the ESB provides several utilities for SOA execution like routing, transformation, mediation, and security.

We start with service orchestration models provided by business analysts. Then, we develop feature models similar to those of SPLE [1] to model commonality and variability requirements. Then, we develop Mediation modules to model the variability requirements of applications. Mediation modules reside in the ESB layer which is above the services layer.

The paper is structured as follows: section 2 presents the ESB mediation modules that are used to realize variability, section 3 details the proposed solution using a running example, section 4 presents work, and section 5 concludes the paper.

## 2. Service Variability Types and Mediation Modules

Service variability can occur in several situations. The following list describes three types of variability based on [3]:

- Composition Variability – where a business process has to select a service from a pool of services.
- Interface Variability – where a service candidate is identified, but its interface is slightly different than the required functionality.

- Logic Variability – where a service candidate is identified, but its internal logic needs a slight change.

Mediation modules are middleware components that reside inside the ESB layer [5]. Mediation modules act on service requests before forwarding them to their destinations. The following is a typical list of ESB mediation modules types:

- Custom – are customized based on specific business requirements. For example, a custom mediation module can be developed to add discount pricing logic to a payment service. We use custom mediation modules to model business logic variability and interface variability.
- Routing –used to route service requests to different services based on some selection criteria. We use Routing modules to route service requests to services based on required feature requests.
- Transformation – used for data format transformation. Some services dictate specific data format for incoming messages. Transformation modules can be used to transform incompatible client requests to the required service’s format.

*In our research, we identify service variability types based on SPL feature selection, and then design mediation modules that realize the selected features as explained in the remainder of this paper.*

We use UML stereotypes to model ESB mediation modules using class diagrams. We use the following stereotypes: <<Mediation>>, <<Service Selection>>, <<Business Logic>>, <<Interface>>, and <<Data Transformation>>.

### 3. Problem and Proposed Solution

In this section we state the problem of our research and then we detail the steps of our proposed solution.

#### Problem Statement

How to design and execute variable service oriented applications independent of the service layer and yet use provided services in a way that satisfies multiple variable scenarios and clients?

#### Proposed Solution

The following paragraphs explain the steps of the proposed solution. These steps will be performed using tool support which is not discussed in this paper.

Service orchestration is modeled by simple activity diagrams where each activity represents a service.

The following diagram depicts a typical orchestration for an E-Commerce Ordering example [7]. We use the <<Service>> stereotype to indicate services and we call this model the ‘Service View’.

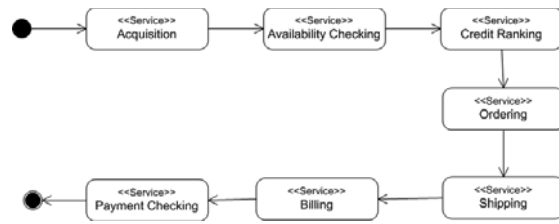


Fig. 1. Ordering Service Orchestration

Features [1] are reusable requirements that are present in SPL members based on members’ specification. All members of the product line have common features. Some members have optional features, whereas other members have to choose from alternative features.

We represent feature models using class diagrams as presented in [1]. The following diagram depicts the feature model for the E-Commerce Order example, which we call the ‘Feature View’

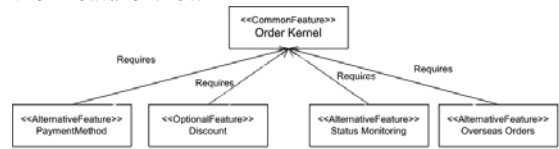


Fig. 2. UML Feature Model for Ordering Example

The Order Kernel feature represents the original service orchestration provided by the ‘Service View’ above. Kernel features are mapped 1-1 to the requested services. The Order SPL has an alternative payment method feature where payments could be performed using credit or debit cards. In addition, there is an optional feature where Discount functionality could be selected. Order status monitoring and overseas shipping could also be optionally selected.

It should be noted that unlike SPLE in [1], kernel features here represent the original service orchestration, but not the services present in all members of the SPL. Instead, product line’s members are made of some services from the original orchestration in addition to ESB modules that realize variability of the remaining services in the orchestration n as explained below.

Optional and alternative features are associated with affected services. For example, the following services are affected by the Discount [7] optional feature: Acquisition, Credit Ranking, Billing, and Payment.

The following diagram depicts the mapping between the Discount feature and affected services:

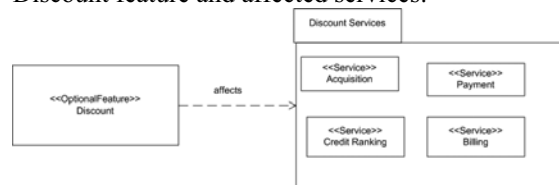


Fig. 3. Discount Feature and Affected Services

Map Alternative/Optional Features into ESB Modules.

- Identify type of service variability (listed above).

- Build, Generate, or Reuse ESB modules.
- Map ESB modules to selected features.

To realize the Discount feature, the following types of services variability and mediation modules are identified:

TABLE 1. Type of Variability & ESB Modules

Service	Type of Variability	Mediation Module
Acquisition	<b>Interface</b> where an extra method is needed for discount inquiry.	<<Interface>>
Credit Ranking	<b>Business Logic</b> where credit ranking is adapted for discount.	<<Business Logic>>
Billing	<b>Business Logic</b> where price is adapted to for discount.	<<Business Logic>>
Payment Checking	<b>Business Logic</b> variability	<<Business Logic>>

The following diagram depicts the mapping between the Discount feature and mediation modules .

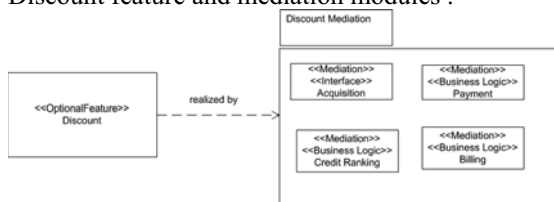


Fig. 4. Discount Feature & Mediation Modules

Compose Kernel, Alternative, and Optional Features.

Like SPL Application Engineering [1], service oriented member applications will be composed using tool support in the following manner:

- Based on feature selection, the desired member application will be composed based on the metadata that was saved in the registry. For example, if the Discount feature is selected, the tool will search the SOA Registry for the mediation modules that realize the Discount feature. These modules were depicted in figure 4 above.
- Mediation modules are then deployed into the ESB of the SOA environment using the environment's configuration tools.
- The tool starts with the original service orchestration depicted in figure 2 above and insert the corresponding mediation activities in place of *affected* services. The following diagram depicts the composed member application based on the Discount feature:



Fig. 5. Discount Feature Member

Notice, that the above activity diagram has activities that represent services <<Service>> and activities that represent mediation modules <<Mediation>>. Activities with the

<<Service>> stereotype represent services from the original orchestrations that were NOT affected by the Discount feature. The sequencing of the orchestration is preserved from the original business process shown in figure 1.

The resulting member application composition could then be expressed in the Business Process Execution Language (BPEL) [6] or any other workflow language.

Design and implement a Receptor <<Receptor>> ESB module for the SPL. This module resides in the ESB and intercepts messages destined for the services participating in the SPL. The Receptor module is based on basic ESB Routing capabilities [5] explained above.

Request messages will have information that indicates the specific SPL and the required features. Message enriching patterns [4] could be used to augment messages based on features selection. The following XML message depicts a conceptual SOAP request that has feature and SPL information in the header part of the message:

```

<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <tns:SPL>Order</tns:SPL>
    <tns:FEATURE>Discount</tns:FEATURE>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

In the above message, we use an XML tag that indicates the name of the SPL <SPL> and another tag that indicates the requested feature <FEATURE>.

Using any SOA environment, member applications could be executed:

- Service requests originating from BPEL code and enriched with metadata indicating the SPL and requested features are sent to the ESB.
- Requests are intercepted by the ESB's Receptor module.
- The Receptor will route the request to the appropriate ESB mediation module.
- The ESB module does its processing and forwards the request to the corresponding service

The following diagram illustrates the execution dynamics for the Discount member application:

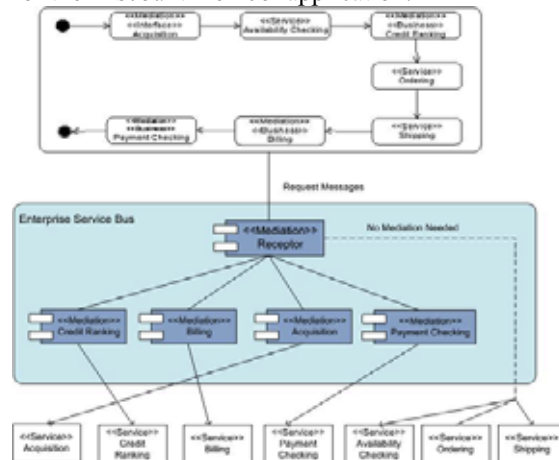


Fig. 6. Discount Member Application

#### 4. Related work

There have been several approaches for designing adaptable service oriented applications. The authors in [8] add variability analysis to an existing service oriented analysis and design method (SOAD). Decision tables are used in [8] to record variability types in each phase of the SOAD process.

The authors in [9] present architectural pattern approaches to model variation points in Web Services.

The authors in [10] advocate a SPL approach that has a specific phase for service composition in the architecture. They introduce several variation points that can be used to customize the SPL during service selection in the application engineering phase. However, the authors in [10] do not tie service selection to the features required in the SPL.

The authors in [11] presented an SPL engineering approach based on Web Services. Components in the architecture were modeled using the <<web service>> stereotype. UI feature selection determined the selection of Web Services. UML activity diagrams modeled customization choices based on feature/Web Service interactions.

In [9], the authors used the concept of features to solve variability problems for SOA. The authors used Feature Oriented Programming to modify the source code of services. However, the authors' approach assumes the availability of service implementation code, which is not the norm in most SOA scenarios.

The authors in [3] presented four types of service variability (Workflow, Composition, Interface, and Logic) and described an approach to adapt services using the ESB. The approach in [3] is similar to our approach in that it uses ESB mediation modules (called Adaptation Managers) to realize variability concerns. However, the approach in [3] does not apply SPL principles, nor does it use features to express variability.

#### 5. Conclusion

One of the major benefits of SOA is the flexible building of IT solutions that can react to changing business requirements quickly. SOA promises a world of IT ecosystem where service providers offer their services and service requesters use these services based on their business needs. Normally, service providers are not aware of service requesters and services are consumed by several requesters with varying requirements.

A research challenge is how to design and execute variable service oriented applications independent of the service layer and yet consume provided services in a way that satisfies multiple variable scenarios and clients.

In this paper, we used SPL variability modeling to model variability of service oriented applications using the ESB layer. We started with service orchestration models and developed feature models to model commonality and variability requirements. Then, we developed ESB

mediation modules to model the variability requirements of service oriented applications.

We believe that our approach has several benefits:

- The same service could participate in many SPLs.
- Services are not aware of variability concerns.
- ESB mediation modules can be updated or added at runtime, thus an SPL can evolve by just manipulating mediation modules.
- Reuse is maximized in SOA based applications by indirectly extending services functionality.
- There is no need to introduce an additional computing layer, since the ESB is present in SOA environments.

We plan to create metamodels that describe the relation between feature and mediation modules. We will take a model driven approach where we consider our 'Mediation View' as a Platform Independent Model (PIM) and the ESB execution environment as the Platform Specific Model (PSM). This approach enables us to devise automated composition of SOA application.

We intend to use commercial and open-source SOA environments to demonstrate the viability of our solution.

#### 6. References

- [1] Gomaa, H. 2005. Designing Software Product Lines with UML. Boston: Addison-Wesley.
- [2] Erl, T. 2005. Service-Oriented Architecture Concepts, Technology, and Design. Prentice Hall.
- [3] Hyun Jung La et al., "Practical Methods for Adapting Services Using Enterprise Service Bus," *Proc. 7th International Conference (ICWE 2007)*, LNCS 4607, Springer-Verlag, 2007, pp. 53-58.
- [4] Bin Wu et al., "Dynamic Reliable Service Routing in Enterprise Service Bus," pp. 349-354, IEEE Asia-Pacific Services Computing Conference 2008.
- [5] Chappell, D. 2004. Enterprise Service Bus. O'Reilly.
- [6] Business Process Execution Language for Web WS, <http://www128.ibm.com/developerworks/library/specification/ws-bpel/>
- [7] S. Apel, C. Kaestner, and C. Lengauer, "Research challenges in the tension between features and services," in *SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments*. ACM, 2008, pp. 53-58.
- [8] Soo Ho Chang, Soo Dong Kim, "A Service-Oriented Analysis and Design Approach to Developing Adaptable Services," pp. 204-211, IEEE International Conference on Services Computing (SCC 2007), 2007
- [9] N. Y. Topaloglu, R. Capilla, "Modeling the Variability of Web Services from a Pattern Point of View". European Conference on Web Services (ECOWS2004), LNCS Springer-Verlag, 2004, pp. 128-138.
- [10] Rafael Capilla, N. Yasemin Topaloglu, "Product Lines for Supporting the Composition and Evolution of Service Oriented Applications," pp. 53-56, Eighth International Workshop on Principles of Software Evolution (IWPSE'05), 2005
- [11] Gomaa, H.; Saleh, M., "Software product line engineering for Web services and UML," *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*, vol., no.pp. 110-, 2005

# Pedagogy-Oriented Software Modeling and Simulation of Component-Based Physical Systems

Dan Tappan  
Department of Computer Science  
Idaho State University  
921 S. 8th Ave., Stop 8060  
Pocatello, ID 83209-8060

*Abstract*—When engineers design component-based physical systems, they have a solid grounding to the real world, which often provides an intuitive reality check about the validity, expected performance, and elegance of their solutions. Software engineers, modeling such systems in the virtual world of computers, lack this insight. Their models frequently bear little resemblance to the real-world counterparts, which undermines the understandability and usefulness. Students develop these disconnected habits because most programming assignments do not enforce a disciplined philosophy for the design and implementation. Furthermore, they have little, if any, opportunity to analyze their typically hacked solutions. This system provides a rich pedagogical framework to help students truly understand what they are modeling and why. In particular, it forces them to separate what a solution is from how it works by promoting established software design patterns and providing a wide range of support infrastructure. It is also heavily oriented toward analysis aspects like testing and performance evaluation. To this end, it seamlessly integrates solutions within a Monte Carlo simulation and a three-dimensional, dynamic, interactive visualization of how solutions would look and act in the real world.

## INTRODUCTION

Whether students realize it or not, a substantial part of programming involves modeling the counterparts of physical systems in the real world. Engineers have the advantage that their work often has concrete, visible aspects, where they can often intuitively say whether a solution looks reasonable [1]. Computer scientists and software engineers, working on systems no less complicated, unfortunately lack this reality check. The result is often that their computational models bear little resemblance to the real world, thereby undermining the usefulness. If they could see a manifestation of how their software would look and behave in reality, they would probably have a better understanding of how to design and implement it more appropriately.

This pedagogy-oriented modeling-and-simulation system provides a comprehensive framework of highly extensible support components and functionality to investigate many concepts and strategies in component-based software systems, particularly involving agents. The overarching design philosophy is to provide support for building and graphically observing arbitrary agents that faithfully model the real world and map back to it. It also helps foster an understanding of proper methodology in the design, implementation, and

evaluation of formal experiments that help determine the effectiveness and efficiency of solutions [2,3,4].

In particular, this system tries to balance the synthesis of creating solutions with the analysis of formally understanding their performance. Most classroom assignments are heavily skewed toward the former, at the expense of the latter; i.e., understanding how a solution works, how well it works, how it might be improved, and so on [5]. Learning is an iterative process, but without the analysis component, a substantial part of the feedback loop is missing [6].

## BACKGROUND

Decades of work in software engineering have identified many aspects of design that contribute to good solutions. Of background interest here, in particular, is component-based software engineering for modeling the compositional nature of systems [7,8]. Aspect-oriented, subject-oriented, and role-oriented software development, as well as object role modeling, align with the philosophy that components should play a precisely defined role—nothing more, nothing less—that the programmer needs to understand and control with respect to real-world counterparts [9,10,11]. Separation of concerns, along with the sound principles of object-oriented design like encapsulation, cohesion, and coupling, focus on the appropriate assembly of discrete components to form functional systems [12,13].

The pedigree of this system attests to the success of its philosophy. It derives—as a complete redesign and reimplementations—from two versions of a large-scale, agent-based modeling-and-simulation system developed for the U.S. Army in support of its Future Combat Systems program [14,15]. In the first version, the inexperienced team produced a monumental (yet admittedly successful) hack. The second version, built within this rigorous framework and philosophy, resulted in multiple awards and the first accreditation of such an analysis tool by the Army in over 20 years. The code was good, the agents were good, and the team learned how and why they succeeded.

This system is written entirely in Java, with Java 3D for the visualization, and JavaCC for the parsing of support files. XML is the standard format for almost all external data. These choices support consistency and portability in deployment and interaction with other tools.



## MODELING

Modeling here refers to defining the composition and communication behavior of software agents that reside within an operational environment. It encourages disciplined forethought in this organization, which the system later strictly enforces to ensure that the agents play by these rules. Similarly, it discourages—and makes obvious—questionable implementation practices (kludges) that deal with issues that should have been addressed at design time.

Arbitrary, component-based agents are the core of the model. The system uniformly accommodates both artificial and natural variants like machines and animals. This approach broadens its usability to cover both well-defined engineering tasks and loosely defined artificial-intelligence tasks. The underlying behavior of agents, and its implementation, are entirely the choice of the student. The system supports these choices by providing a flexible modeling framework that helps students separate *what* a solution is supposed to do (and not do) from *how* it does it [1,16]. In particular, it takes advantage of well-established design patterns that divide agents into their structural, behavioral, and creational elements [17]. It also provides a wealth of miscellaneous supporting functionality that students do not need to implement themselves.

### A. Structural Elements

Structural elements define the composition of agents, and by extension, systems of agents, in terms of interconnected components [16]. Physical components are literally the building blocks of an agent that has a concrete, real-world counterpart. For visualization purposes, every component is abstracted as a rectilinear box defined by a unique identifier and width, depth, and height dimensions. Fig. 1 (a) shows a simple tank consisting of a hull, turret, barrel, and sensor. Various appearance attributes, such as color, transparency, and solid and wireframe representations, are also available. For higher-fidelity visualization, the box may also contain an externally defined three-dimensional model. Fig. 1 (b) shows an elephant with three attached sensor components. The system currently supports Wavefront™ and 3D Studio™ models, many of which are freely available on the web. Google also provides a vast library of compatible models for its SketchUp™ software, which is available for free evaluation. Although these models can be arbitrarily complex polygons, the system still bounds them as boxes because, from a software perspective, their geometries are irrelevant.

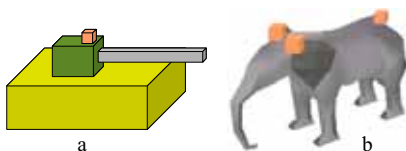


Fig. 1: Compositional Agents

In a good design, each component should have a real-world counterpart, but it may not have a definite physical manifestation; e.g., a logger. These virtual components are still encapsulated within other components, so it is equally important to visualize them as part of the overall

compositional structure. Therefore, they also require dimensions, which are arbitrary.

The recursive interconnection of components is based on the concept of an engineering gimbal. This formalism requires that each component have a socket positioned somewhere relative to the origin of the component. It may also have constraints that define and limit its degrees of freedom (DOF), as well as advanced aspects like latency, speed, and acceleration. The socket allows another component (the supercomponent) to mount this component (its subcomponent) to itself through a ball, which is somewhere relative to the origin of the supercomponent. A component can have any number of balls, allowing for unbounded compositionality and degrees of freedom.

Managing these essential physicalities invariably causes students tremendous grief. Any practical implementation (such as the quaternions used here) is decidedly nontrivial and far beyond the scope of most programming assignments. To mitigate this problem, a separate DOF manager for each component provides a wealth of features, for which students merely need to define the constraints. This declarative approach frees them to focus on what the interconnections are supposed to do and why, and not so much on how their code needs to make them operate. The manager supports the two common DOF systems in Fig. 2, which reflect how most real-world components articulate.

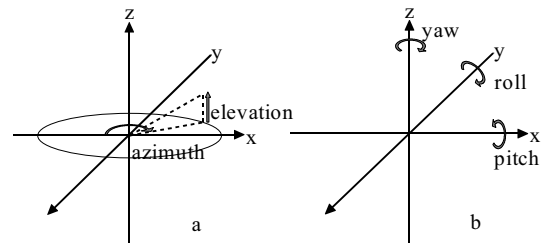


Fig. 2: Degree-of-Freedom Systems

The 5-DOF system (a) uses azimuth and elevation for components like a gun barrel, whereas the 6-DOF (b) uses yaw, pitch, and roll for components like a shoulder joint. In both cases, the three degrees of freedom for position ( $x$ ,  $y$ , and  $z$ ) are identical. The choice of system is important because student code needs to know how to manipulate it properly. In the 5-DOF system, each degree of attitude freedom is independent of the other, and either of the two possible orders for changes results in the same configuration. In the 6-DOF system, however, there are six possible orders, each of which may result in a different configuration. The importance of such design considerations in a physical agent is normally lost on students, but this approach brings it to the forefront.

The gimbal system manages dependencies between components. Each agent has a required base component, such as the hull of the tank in Fig. 1 (a), as well as optional subcomponents, like the turret on the hull, which in turn has sensor and gun subcomponents. Changing the configuration of any degree of freedom on any component automatically propagates the corresponding changes and dependencies throughout all subcomponents.

As a reality check, this process also verifies that student code does not force the mechanical system into an invalid configuration. For most systems, the behavior of the code and the real world must correspond (within the design abstractions and limitations). However, it is very easy and common for code to perform an action that would be obviously invalid or even impossible in real life. Unfortunately, within the virtual perspective of software, this result is often not apparent, especially if the programmer does not truly know what he or she is modeling. The mantra of this system is that components should do what they are supposed to do, and not do what they are not supposed to do. The former is usually self-evident even to the most inexperienced programmers, but the latter tends to elude even professionals. A solid grounding in the reality of what is being modeled is essential.

Sensors are a very common component in most agent-based systems [18]. They acquire information from the environment and pass it to other components for processing and subsequent actions. The fixed variant is statically dependent on the spatial configuration of its supercomponent; e.g., a camera mounted on a car. The movable variant has additional dynamic independence; e.g., a camera with a pivot platform mounted on a car. Both variants support a horizontal, vertical, or combined field of view, which helps determine whether another agent or component is within an angular wedge (for two dimensions) or a pyramidal frustum (for three dimensions), as in Fig. 3 [19]. Movable sensors can adjust the field of view within the angular limits of a field of regard to support scanning; e.g., panning the camera plus or minus 45 degrees from center. The timing subsystem can automatically manage this movement, or student code can perform it manually.

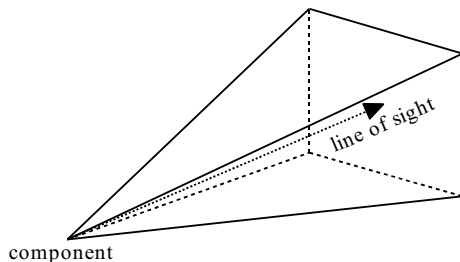


Figure 3: Frustum

### B. Behavioral Elements

Behavioral elements overlay the structural elements. A student must first define *what* an agent is, then *how* the agent functions within these physical constraints. This philosophy enforces a prescribed chain of responsibility for delegating the execution of requests to the appropriate components within an agent or between agents. It discourages undisciplined programmatic “cheating,” where the model uses information or performs actions that it rightfully should not be able to access. For example, when a sensor is to process visual percepts to determine where another agent is, it is easy (intentionally or not) in code to access another programmatic object through direct method calls to get the actual data, as opposed to indirectly deriving the believed data. This shortcut probably undermines the goal of modeling the sensor, though.

While behavioral elements cannot prevent students from implementing dubious solutions, they can at least make such approaches more apparent for code review, grading, and follow-on discussion. The overarching design philosophy of this system is, after all, to build agents that faithfully model the real world and map back to it. If code can do something that is not possible in the real world, then it needs to be examined and justified carefully. Students need to learn firsthand what is justifiable—and how to justify it persuasively—because most design decisions in computer science are open-ended and subjective [20,18].

Behavioral elements are a conglomeration of at least the composite, strategy, chain of responsibility, mediator, state, observer, command, and interpreter design patterns [17]. The code itself serves as a strong metaexample of an architecture with a disciplined design. The core element provides a communication network with formal protocols for the intra-agent and inter-agent transfer of messages, which are typically events or requests for data or actions. These protocols greatly reduce the complexity of managing intercommunication, while also reducing the temptation for programmatic cheating and outright hacking at a solution. Components are forced to communicate through messaging only; no direct method calls are allowed. This approach admittedly introduces significant performance limitations, but in a pedagogical environment, the disciplined learning experience is much more important. Furthermore, it allows the system to log, trace, and visualize all communication, which truly reveals (for better or worse) the inner workings of student code.

A message is an envelope with the sender and recipient identifiers, as well as the arbitrary payload to deliver. Recipients can be individual components, classes of components (e.g., all agents in a group), or general broadcasts to all components. A message allows itself to be handled in various ways (abridged here) depending on the recipient and its state:

- **IGNORED:** a recipient determines that a message (usually a broadcast) is not intended for it or is irrelevant, so it silently discards it.
- **ACCEPTED\_PROCEED:** a recipient determines that it can immediately act on a message and return any result. The sender can proceed after receiving it.
- **ACCEPTED\_WAIT:** a recipient determines that it can act on a message, but the result will be forthcoming after an indeterminate delay. A confirmation of this condition is returned in the interim so the sender knows to expect the result eventually, and it can proceed with other activities (if possible) until then. It can also cancel the message and reissue it elsewhere.
- **REJECTED\_DISCARD\_INFORM:** the recipient would otherwise be able to act on a message, but it cannot at the moment. This condition is returned to the sender, which can then decide how to handle it. A **REJECTED\_DISCARD\_SILENT** variant omits the return, which is similar to instructing the post office to abandon a package that cannot be delivered to avoid the return postage.

- `REJECTED_RESUBMIT_INFORM`: the recipient would otherwise be able to act on a message as described above. It will inform the sender when it is free to accept the message again, at which time the sender must decide whether to reissue it. In the meantime, it could also reissue it elsewhere.
- `REJECTED_RESUBMIT_AUTO`: the recipient would otherwise be able to act on a message as described above. It holds onto the message until it can process it, at which time it informs the sender of either `ACCEPTED` state.

Another useful aspect of messaging is its novel extension to crisp boolean logic. In addition to `true` and `false`, there is `unknown`, which indicates no commitment to either of the other two yet. If the execution of an action is dependent on a conditional evaluation with an unknown state, it probably cannot occur at this time. This task can be scheduled or deferred in a number of ways; for instance:

- If subsequent code is dependent on this condition, a block can occur until the unknown state is resolved through any of the protocols above.
- If subsequent code is not dependent on this condition, execution may continue in parallel while the unknown state is resolved. A message will interrupt it when the answer is finally available.

The messaging system effectively emulates many aspects of multithreading. It does so deterministically, however, to guarantee that behaviors are predictable. This aspect is critical for stochastic simulation because undesirable nondeterministic behavior can undermine inferred cause-and-effect relationships in evaluation [21].

Another convenient behavioral element supports timing and synchronization for events, which can be overwhelming programming tasks, especially for stochastic, multiagent simulations. This service allows components (and their DOF managers) to subscribe to updates. It allows for finer-grained actions than would otherwise be reasonably available. For example, slewing the azimuth of a sensor from 0 to 90 degrees over five seconds would require substantial student code in an agent to account for the aspects of time and reasonably smooth movement. Inconsistent approaches often lead to bizarre “relativistic” effects where multiple agents operate under different time systems. Time is a shared phenomenon that is best handled globally by the system, not locally by the agents. In this case, the operational approach to slewing must also be carefully considered. If a student simply increments the angle by 90 degrees after a five-second wait, at no time does the sensor pass through any intermediate angles, which may result in its not seeing another agent. The timing subsystem allows subscribers to specify in intuitive ways how they want to be updated; for instance:

- $n$  times over  $s$  seconds
- every  $s$  seconds over  $t$  seconds
- every  $s$  seconds  $n$  times
- every  $s$  seconds continuously until canceled
- one time for  $s$  seconds, then self-cancels

In this example, the sensor component could subscribe to be updated nine times over five seconds uniformly. For each update, it can increment its azimuth by 10 degrees. In this way, time is appropriately delegated to the system, while space (movement) belongs to the sensor. Such disciplined responsibility, cohesion, and coupling are key to effective, understandable software [9]. Inappropriate attempts at localized management can be a nightmare to debug and maintain.

Other background processes manage specialized operations. Collision detection, for example, is expensive and often unnecessary. Only components that want to participate should subscribe.

### C. Creational Elements

Creational elements manage the existence of components, namely their creation, assembly, disassembly, and destruction. They use prototype, flyweight, factory, and builder patterns to simplify this process so students just need to ask for an agent shell configured as they want (within its specified design limitations) [17]. Agents can be permanent, such as walls, semi-permanent, such as spaceships in a game, or temporary, such as photon torpedoes emanating from one spaceship and expiring after a collision or a certain range or time (which is managed by a behavioral service). The XML definition of the composition of an agent is analogous to the data definition of a class in object-oriented programming, and an agent is likewise an instance of it. Instantiation of components is mostly a rubber-stamp process with minor variation (e.g., identifiers), not unbounded as with ordinary constructor calls. This philosophy forces student implementations to conform to their designs. It especially limits the temptation to throw more parameters at code, which greatly increases the coupling and subsequent disorganization.

### D. Support Elements

Roughly 30 support elements provide a variety of miscellaneous helper functionality for modeling common aspects of the real world [22,19]. While these elements might not necessarily be difficult to implement *per se*, anecdotal evidence overwhelmingly supports the conclusion that students spend an inordinate amount of time on such peripheral aspects of their assignments, at the expense of the intended content. Some examples are the management of movement and coordinates, like points, lines, planes, boxes, and paths, and common algebraic, trigonometric, engineering, and physics functions. All derive from a common data type that manages some of the contextual difficulty in their usage. Common conversions for units, magnitudes, and between types are provided, and the algebraic validity of results can be verified in many cases.

## SIMULATION

Simulation here refers to putting the agents into operational scenarios within an environment to observe and measure their performance with respect to well-defined tasks. It facilitates a process of rapid, iterative refinement, in which a student runs a solution, observes how it performs, and improves it. The simulation provides a Monte Carlo framework for controlled

experiments that, if properly constructed, can convincingly demonstrate improvements and overall performance [23].

The control simulation establishes baseline performance. A student initially runs his or her solution to see what it does. In a typical educational setting, this step is as far as the process usually goes. In other words, at some point, most students subjectively and arbitrarily decide that their solution is “good enough” and simply stop. There is rarely an objective measure of performance; nor does this strategy help them to learn how to establish one. Most of the effort is spent on synthesis, and any analysis is minimal.

The test simulation measures differences in performance with respect to the control simulation. For example, an agent is supposed to find another agent with its sensors. The control simulation, using one type of sensor, has a certain quantitative performance, but no benchmark against which to assess its meaning or significance. The test simulation, using a different type of sensor, and differing *only* in this respect, results in different performance. The difference between the two can be attributed directly to this one and only change. If the change was an improvement in performance, then it suggests a trajectory toward further improvement. Likewise, a negative or inconsequential change suggests trying a different approach. This process of continual refinement allows students to see the cause-and-effect relationships of their choices immediately.

The components are carefully controlled within the simulation framework to ensure that modifying or swapping them has no side effects, or at very least, that the side effects are known. This stochastic stability is critical because having more than one change may confound the analysis and undermine any conclusions. The complexity and difficulty of this task would preclude any reasonable expectation that students could properly code it themselves.

## VISUALIZATION AND ANALYSIS

The execution of agents is interesting and often entertaining to watch. Observation is also a useful, qualitative measure of performance: a student may be able to see immediately whether an approach converges toward the expected solution. To this end, a basic, three-dimensional, interactive visualizer is available to render a simulation graphically. Fig. 4 shows the behavior of an agent following a path while enforcing certain movement constraints.

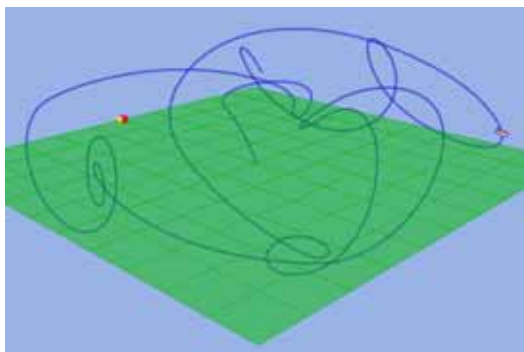


Fig. 4: Visualization

Various metainformation is selectable to indicate identifiers, positions, attitudes, velocities, paths, and so on. Furthermore, a graphical map can show the underlying interconnections between components and agents, on top of which it can visually depict message traffic. Selecting pieces of it exposes internal details like sender, recipient, payload, protocol, and creation time in detailed text form. This capability is invaluable for determining how a system is operating, and where performance bottlenecks and potential points of failure are, and so on.

The simulation can be played like a movie, but dynamically from any vantage point. Visualization facilitates classroom demonstrations by instructors and students. In combination with written reports and small presentations, this feature can help students improve their communication skills. It also helps instructors because students can contribute to the evaluation and relative grading of each other's performance. Finally, it serves as an appealing demonstration for computer-science recruitment events, which often lack a “coolness” factor to attract interest [24].

For quantitative analysis, nearly all internal data can be exported through the logging subsystem to common applications like Excel™, MATLAB™, and gnuplot in their native formats. Determining which data to capture and their significance is, of course, the students' responsibility.

## EVALUATION

Formally evaluating a modeling-and-simulation, test-and-evaluation system like this one is a somewhat circular process. A reasonable, intermediate measure of success is how well it has integrated into a number of undergraduate design courses so far as a proof of concept. In particular, it has been the core software for several offerings of a lower-division course in advanced object-oriented design and an upper-division course in software engineering. Under a different guise, it has also contributed to two offerings of an upper-division course in artificial intelligence and control systems, with a total of 16 assignments on topics like autonomous vehicle navigation and robotic-arm manipulation.

Ultimately, it is the students who decide whether any software helps or hinders their educational experience. In this case, anecdotal evidence and formal course evaluations suggest that it has been received well. Significant work is still needed to polish it for widespread distribution. Nevertheless, it has shown itself to be a worthwhile contribution to design-related pedagogy for multiple curricular emphases.

## FUTURE WORK

There are two facets to future work with this system. Within the existing framework, a friendly, unified graphical user interface is necessary. The underlying organization of XML files for data and configuration is currently somewhat complex and onerous to set up. This limitation conflicts with the philosophy of a rapid-prototyping system.

Beyond the existing framework are spin-off applications. The current system is intended for the university setting. It would be excessively complicated and overwhelming for

younger audiences, who could also benefit from it, however, because they tend to show great interest building and programming robots, for example [24]. One spin-off is a simplified facade around the system, such that the basics of defining and running an agent are easily accessible. Fun and logical thought would be the focus, not formal evaluation. However, for advanced children, especially those participating in competitions like the FIRST Lego League robotics challenges, some analysis could be practical. For example, there are many options in deciding on the kind of robot to build. In practice, teams tend to fixate on one and build only it. If, through some analysis appropriate to their level, some designs appear more promising than others, they could target their effort more effectively.

## CONCLUSION

This system provides a powerful software environment with a large tool set of commonly needed features for designing, implementing, testing, and evaluating models for component-based software systems and engineering problems. Its goal is to mitigate much of the tedious, time-consuming, error-prone administrative aspects to programming solutions, so students can focus more on the task and less on the support code. It also promotes careful design and analysis by requiring forethought before implementation, and by enforcing constraints. In various incarnations, the system has been successfully fielded over several semesters for courses in object-oriented design, software engineering, and artificial intelligence. It has been well-received and shows promise as a publicly available package for others to use eventually.

## REFERENCES

- [1] P. Denning and R. Riehle. 2009. The Profession of IT: Is Software Engineering Engineering? Communications of the ACM, March, Vol. 52, No. 3.
- [2] Z. Dilli, N. Goldsman, J. Schmidt, L. Harper, and S. Marcus. 2002. A New Pedagogy in Electrical and Computer Engineering: An experiential and conceptual approach. In *Proc. Frontiers in Education FIE02*.
- [3] T. Wiesner and W. Lan. 2004. Comparison of Student Learning in Physical and Simulated Unit Operations Experiments, *Journal of Engineering Education*, July.
- [4] L. Tong. 2003. Identifying essential learning skills in students' engineering education. In *Proc. HERDSA 2003*, Canterbury, New Zealand.
- [5] D. Tappan. 2009. ShelbySim: A Transparent, Pedagogy-Oriented Simulator for Computer-Based Systems, *International Journal of Engineering Education*, in press.
- [6] R. Irish. 1999. Engineering Thinking: Using Benjamin Bloom and William Perry to Design Assignments, *Language and Learning Across the Disciplines*, Vol. 3, No. 2, pp. 83-102.
- [7] G. Heineman and W. Council. 2001. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley, Reading, MA.
- [8] C. Szyperski. 2002. *Component Software: Beyond Object-Oriented Programming*. 2nd ed. Addison-Wesley, Boston.
- [9] R. Filman, T. Elrad, S. Clarke, and M. Aksit. 2007. Aspect-Oriented Software Development, in *Proc. 5th Workshop on Software Engineering Properties of Languages and Aspect Technologies*, Vancouver, BC.
- [10] S. Clarke and E. Baniassad. 2005. *Aspect-Oriented Analysis and Design: The Theme Approach*. Addison-Wesley, Boston.
- [11] D. Parnas. 1972. On the Criteria To Be Used in Decomposing Systems into Modules, *Communications of the ACM*, December, Vol. 15, No. 12, pp. 1053-1058.
- [12] P. Tarr, H. Ossher, W. Harrison, and S. Sutton, Jr. 1999. N Degrees of Separation: Multi-Dimensional Separation of Concerns, in *Proc. International Conference on Software Engineering*, May.
- [13] M. Aksit, B. Tekinerdogan, and L. Bergmans. 2001. The Six Concerns for Separation of Concerns, in *Proc. ECOOP Workshop on Advanced Separation of Concerns*, Budapest.
- [14] J. Engle. 2005. AMSAA's SURVIVE Model plays key role. *RDECOM Magazine*, August.
- [15] D. Tappan and J. Engle. 2005. The AMSAA SURVIVE Model. In *Proc. U.S. Army 16th Annual Ground Vehicle Survivability Symposium*, Monterey, CA.
- [16] M. Jones. 2008. *Artificial Intelligence: A Systems Approach*. Infinity Science: Hingham, MA.
- [17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Indianapolis, IN.
- [18] S. Russell and P. Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Pearson, Upper Saddle River, NJ.
- [19] D. Bourg and G. Seemann. 2004. *AI for Game Developers*. O'Reilly, Sebastopol, CA.
- [20] R. Pressman. 2010. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York.
- [21] H. Gould and J. Tobochnik. 1988. *An Introduction to Computer Simulation Methods, Part 2, Applications to Physical Systems*. Reading: Addison-Wesley.
- [22] D. Bourg. 2002. *Physics for Game Developers*. O'Reilly, Sebastopol, CA.
- [23] G. Fishman. 1995. *Monte Carlo: Concepts, Algorithms, and Applications*. New York: Springer.
- [24] P. Jonsson. 2008. "Can competitions raise 'cool' factor of math, science?" *Christian Science Monitor*, 17 May.

# An Academia-Industry Collaborative Teaching and Learning Model for Software Engineering Education

Huilin Ye

*School of Electrical Engineering and Computer Science  
The University of Newcastle, Callaghan, NSW 2308, Australia*  
[Huilin.Ye@newcastle.edu.au](mailto:Huilin.Ye@newcastle.edu.au)

## Abstract

*Teaching and learning environment limited only to classrooms and labs at universities may not suffice for software engineering students to develop practical and professional skills that are central to students' future roles as industry professionals. It is important to introduce the state of the arts industry professional practice into software engineering education program. An academia-industry collaborative teaching and learning model has been developed and employed in a capstone software engineering course for a sustained period. A standard software development methodology, called Process MeNiOR, provided by the industry collaborator has been used to guide student project development. A set of teaching and learning strategies with strong industry involvement has been employed to inspire student learning enthusiasm, to develop student practical professional skills. As a result, student career prospects have been enhanced. Formal student surveys, informal student and industry feedbacks have demonstrated these achievements.*

## 1. Introduction

Software is now ubiquitous – embedded in nearly every aspect of modern society, both personal and professional. Businesses and the society as a whole run on software. The increasingly global business climate has accelerated the need for business software. The software industry has experienced great challenges in meeting this demand. There is a growing shortage of software engineers in Australia and worldwide. As software engineering educators we must prepare our software engineering students for the challenges confronted by the industry, and for them to work effectively in industry after graduation.

Software engineering is an emerging engineering discipline that applies a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software [1]. Software engineering focuses on the study of methodologies that support systematic, productive, and robust software developments. The principles of these methodologies may not be difficult to understand. However, the effective application of these methodologies in real-world software project development

processes is difficult, especially for novices. It is therefore important to introduce software industry professional practices into our software engineering courses to enhance the practical and professional skills of software engineering students. The dual challenges of society's critical dependence on the quality and cost of software, and the relative immaturity of software engineering, make attention to professional practice issues even more important to software engineering programs than many other engineering programs. Graduates of software engineering programs need to arrive in the workplace equipped to meet these challenges and to help evolve the software engineering discipline into a more professional and accepted state [2]. However, teaching and learning approaches limited only to lectures and labs do not provide sufficient support for software engineering students to learn practical and professional skills. It has been recognised that many software engineering graduates are inadequately prepared for industry software development and innovative approaches to teaching and learning should be developed to solve this problem [3].

This imperative has inspired the creation of a systematic academia-industry collaborative teaching and learning model that immerses our software engineering students in the industry-based themes, issues and problems of software development. This has been achieved by embedding a large scale, team-based project based on a world-class industry development methodology as a major assessment item in our capstone software engineering course. These developments were preceded by the identification of a set of core practical and professional skills and attributes required for our students to work effectively in industry after graduation. A set of teaching and learning strategies was then developed to provide students with opportunities to work towards these skills and attributes. This teaching and learning model has successfully motivated and inspired students to learn in ways that support them to develop practical and professional skills and attributes, and has gained recognitions from our software engineering graduates and the software industry.

The remainder of the paper will be organised as follows. Section 2 will provide background information about the teaching and learning model and identify the set of professional skills that are important for software

engineering graduates. Section 3 will discuss a set of teaching and learning strategies employed to achieve the set of professional skills. Section 4 will evaluate the teaching and learning the model. Section 5 will conclude the paper.

## 2. Important Professional Skills for Software Engineering Graduates

The bachelor of software engineering program at our university is a 4 year degree. A third year core course in our software engineering program, called SENG3120 Object Oriented Software Engineering, is selected for employing this academia-industry collaborative teaching and learning model. In the context of the overall program the students' work in SENG3120 functions as a capstone experience given that all the knowledge and skills learned in courses taken previously are linked together to develop a complex, team-based software project. The major problems of software development projects are not so much technical as sociological in nature. The industry seems to agree very much with this statement while the university seems to give it little importance [4]. Some sociological skills are considered more important than technical skills for software engineering graduates to work effectively in industry. Each year, the American National Association of Colleges and Employers conducts a survey to determine what skills employers consider most important in applicants seeking employment. In 2003, employers were asked to rate the importance of candidate qualities and skills on a five-point scale, with five being "extremely important" and one being "not important." Communication skills (4.7), teamwork skills (4.6), interpersonal skills (4.5), motivation and initiative (4.5), were the most desired sociological skills [5]. However, it is difficult to develop these skills using the traditional lecture based teaching and learning model. Industry professional practice should be introduced into our software engineering program to enhance these skills. Furthermore, software products are among the most complex of man-made systems, and software by its very nature has intrinsic, essential properties (e.g., complexity, invisibility, and changeability) that are not easily addressed [6]. The main challenge of software engineering education is to prepare students for the real world software product development, which is inconsistent, complex and always in a state of change [7]. It is important to expose software engineering graduates to the common themes of project work that occur within industry to learn important lessons of real world software development, and consequently to develop their ability of coping with uncertainty, complexity, and changeability.

The industry partner for this academia-industry collaborative teaching and learning model is Object Consulting Pty Ltd (Object). Object is Australia's leader in software development for large-scale business applications. Object has utilised the best industry practice and experience to develop a world-class system development methodology, called Process MeNtOR. This is key to

Object's ability to create robust, reliable, scalable and secure systems and to consistently deliver projects on time and within budget. Object has specifically created a simplified version of MeNtOR that can be used in our software engineering education program. Students in SENG3120 are required to develop team-based software projects following the methodology specified in MeNtOR.

It is important to know what the software industry expects from our SE graduates and what professional skills are required for graduates to work effectively in industry. The following skills, identified by both academia and industry, are considered central to their future role as professionals in the software industry. The students' activities and assessment in SENG3120 crystallises their development of these key skills and attributes.

- proficiency at applying systematic standard methodologies to guide software development processes: the principles of software engineering methodology are not difficult for students to learn. But how to effectively apply these principles into real-world software development is challenging. "A learned concept is much more valuable if we know how to apply it to a real problem" [8]. This is extremely true for software engineering discipline. It is a relatively immature engineering discipline and costly failures of software developments were frequently observed. Introducing the state-of-art industry practice into software engineering program will enable our graduates to develop software products following industry standard methodologies to achieve high quality and productivity in their work.
- an ability to resolve ill-defined situations and problems: software products are intangible in nature and always in a state of change, which makes the initial specifications of software products always ambiguous and incomplete. How to tackle the ill-defined situations and problems is an important lesson for our students to learn.
- working in teams to construct relatively large software projects: modern software products become very large and complex. It is impossible for such a product to be developed by a single person. Team skills are essential and very important in software development. Group work also provides an environment that allows most of the theoretical issues learned in classrooms to be practised [9 - 10].
- understanding of the importance of employing feedback to effect improvements: The changeability of software products makes the software development process iterative. This iterative development process occurs in the software industry all the time and is an unavoidable part of software development. Effectively employing feedbacks to effect improvements is crucial to the success of software development.
- an ability to communicate effectively and professionally within groups and with end customers: communication skills, both written and verbal, were

ranked as the most desired characteristics of university graduates by industry in Australia [11].

### 3. Teaching and Learning Strategies Employed in the Model

This academia-industry collaborative teaching and learning model embeds a large scale, team-based project development based on a world-class industry development methodology in SENG3120. The project given to the students each year is usually a business software application, such as network based security management system, on-line flight booking system etc. The employed teaching and learning strategies are described below.

#### 3.1 Applying MeNtOR in the Project Development Processes

Most of the software development tasks completed by the students before SENG3120 are small pieces of software coding which are executed in relatively ad hoc ways. Without a systematic, professional methodology to direct system analysis and design, simply coding a large software system will never be successful. Process MeNtOR provides such a methodology. It divides the whole development process into three stages. The activities, deliverables, and document templates for each stage are clearly specified. Under the guidelines provided by MeNtOR students have to learn how to partition the whole software development job into manageable tasks; to estimate the time-effort spent on each task; to appropriately assign the tasks to team members; to integrate the components into a complete software application; and finally to deliver the software system on time. By following the Process MeNtOR, students are exposed to the common themes of project work that occur within software industry and gain hands-on experience by go through the development life cycle.

#### 3.2 Develop Students' Abilities to Resolve Ill-Defined Problems

Transforming a set of ill-defined system requirements in a precise and complete set of system specifications is crucial for successful software system development. Every year the students are given a set of software system requirements around which they develop their SENG3120 project. These requirements are purposely ill-defined, ambiguous and incomplete. Students have to learn how to identify ambiguities, collect outstanding information, and apply professional methods to revise the requirements. Students learn that failure to sufficiently define all the necessary details of the requirements results in the final delivery of an incorrect and incomplete software system. They gain the understanding that late corrections of mistakes made during the early stages of software development are very costly as these mistakes will have cascade effects which influence all the following stages. This is a commonly recurring theme in the software industry, and an important lesson for students to learn. Their engagement with clients and brainstorming of

remedial activities with team members also improve their communication and interpersonal skills.

#### 3.3 Fostering Students in Effective Team Organisation and Management

Software development has to be executed by teams in software industry because of the huge amount of effort involved. A SENG3120 project team consists of three or four students. Each team has a team leader who is responsible for keeping log books, regular meeting minutes, and email communication details. These records are used as a basis to address team problems should any occur. The development work is shared by team members, with each member being responsible for a portion of the work. The assembly of the portions into a complete and effective software system is a challenging job as the mistakes made in discrete portions cannot be easily discovered until they are being put together. Students have to learn how to define the interfaces between the portions appropriately; and to work with their team members to make decisions to resolve conflicts.

#### 3.4 Providing feedback and comments to students

The deliverables for the three stage project are listed in Table 1.

Stage	Deliverables
1	Project Plan
	Requirements Model
2	Revised Project Plan and Requirements Model
	System Model
	Component Model
	Test Model
3	Final version of System Model, Component Model, Test Model
	Executable software application and source code

**Table 1: Project Deliverables**

Early stage deliverables must be reviewed in the subsequent stage for revision. In addition to the marks given to the students for each stage's deliverables, detailed feedback and suggestions are made available to them. Any mistakes or inappropriate designs made in earlier stages' deliverables must be corrected or revised. The updated work must be re-submitted as part of their project assessment. This iterative development process occurs in the software industry all the time and is an unavoidable part of software development. Having experienced this iterative process the students understand how to effectively use feedback to improve their projects.

#### 3.5 Inspiring student learning enthusiasm through industry recognition

An annual Object Technology Prize (certificate and \$1500 provided by Object) has been established to sponsor SENG3120 project competitions. Three best teams are selected every year as finalists and they give a presentation and a demonstration of their project at the project competition. A representative from Object Consulting



comes to our university to judge the presented projects and selects one winner for the prize, and then talks to the students about why the winner’s project is the best one. Students greatly appreciate having professional engineers review their work and offer them feedback. The winner, as well as the other finalists, receives a certificate as recognition of the excellence of their work. Object also issues MeNtOR certificates to the students who achieve Distinction or above in both the project and the MeNtOR questions in their SENG3120 final exam. These certificates are recognised by the software industry and the recognition enhances the employment prospects for the students.

### 3.6 Sufficient resources are provided to the students for learning.

Our industry partner provides MeNtOR process as CD-ROMs for students to borrow at our Discipline office. User-manuals about MeNtOR are available on short-loan in the library. Students can also access Object’s web site to gain a more detailed understanding of MeNtOR. To help new SENG3120 students understand the project development, a previous successful student project is demonstrated at the beginning of the semester. Some industry guest lectures given by previous SE graduates were organised to introduce their SENG3120 project experiences to the new SENG3120 students. Every year SENG3120 project competition is open all software engineering students to inspire their interest in the course.

## 4. Impact and Evaluation

The effective employment of this academia-industry collaborative teaching and learning model has inspired student learning enthusiasm, improved student practical professional skills, and enhanced student career perspectives. Formal student surveys, informal student and industry feedbacks demonstrated these achievements. This model has also contributed to the implementation of our academic program objectives.

### 4.1 Inspired student learning enthusiasm and improved practical professional skills

The challenges of a complex project development, the stimulation of the project competition, and the industry recognition of their work have motivated and inspired students to learn. By experiencing first hand important industry lessons through the process of their project development students understand how practical professional skills relate to their software engineering program and their career, and how important these are. The sample survey results shown in Table 2 obtained from Student Evaluation on Teaching demonstrate this understanding. The scores shown in the table are the average scores collected over the last three annual evaluations. The survey also includes a qualitative element that includes a question that asked “what is the best thing about this course?” In response to this question around 85 percent of students considered the real project

development, the experience of group work, and practical training to be the best things about this course.

Student survey questions	Survey result	Implication of the survey results:
The course is relevant to the program	3.95	1: Strongly disagree; 2: Disagree; 3: Neutral; 4: Agree; 5: Strongly agree
The course is relevant to my career	4.0	
The lecturer is professional in attitude	4.0	

**Table 2: Student Survey Results**

**Student feedback:** *It is an understatement to say that this course [SENG3120] was more challenging, more rewarding and achieved a higher level of student enthusiasm than any other course.*

Students’ enthusiasm to study has resulted in excellent student performance in their project. Our software engineering students have excelled in independently solving problems encountered in the development processes and have gained valuable experience from correcting their mistakes and improving their works. Their oral and written communication and interpersonal skills have also been significantly improved by working in teams. Our industry partner has been impressed with the projects presented at the annual project competitions.

**Student feedback:** *The benefit of learning team based development, documentation standards and a formalised software process translate not only into final year projects but also into industry.*

**Industry evaluation:** *We are continually impressed with the quality of the work, and the quality of the presentations at the end of this subject [SENG3120].*

### 4.2 Enhanced career prospects of software engineering graduates

The high quality and competitiveness of our software engineering graduates have also been recognised by the software industry and inspired their interest in recruiting our graduates. For example, Object often sends their job advertisements to our university to recruit our software engineering graduates. Our software engineering graduates employed by Object have been appraised by the company as “excellent”. The valuable SENG3120 project development experiences make our software engineering students more attractive to prospective employers.

**Student feedback:** *I found it was one of the most useful projects to refer to and discuss with prospective employers. I strongly believe that the successful completion of the SENG3120 project made it possible for me to present myself in a way which was attractive to employers.*

**Industry evaluation:** *The success of this course is more than demonstrated in the quality of the students and their preparedness for work in the commercial software development arena.*

### 4.3 Contribution to the implementation of our software engineering program objectives

The successful employment of this teaching model has made significant contribution to the implementation of our software engineering program objectives. Our software engineering program is currently in the professional accreditation process. Our discipline has prepared a document, called Software Engineering Program Objective Matrix, for the accreditation. The document identifies 42 software engineering program objectives that are the professional skills and attributes expected to be achieved in the program. The matrix maps 28 courses offered in our software engineering program to the objectives achieved by the courses. SENG3120 achieved the highest mapping. It has been mapped to 24 program objectives in comparison with the average 8.8 mapping per course. The author has extracted from the document the following sample program objectives mapped to SENG3120 but rarely mapped to other software engineering courses. The teaching and learning activities involved in SENG3120 have certainly contributed to the improvement of these professional skills for our software engineering students.

- Ability to resolve ill-defined situations and problems through the application of their engineering specialisation knowledge, skills and attitudes to the partitioning of a problem and the management of the problem components
- Proficiency at conducting and managing an engineering project to achieve a substantial outcome to professional standards, or as a member of a team conducting such a project, and ability to demonstrate a key contribution to the team effort and the success of the outcome
- Both spoken and written languages to communicate effectively to both professional and non-professional groups.

### 5. Discussion and conclusions

In this paper, we have reported our experience of developing and employing an academia-industry teaching and learning model into university software engineering program. A set of practical professional skills are recognised important to software engineering graduates but the teaching and learning environment limited only to classrooms and labs at universities may not suffice for students to develop these skills. Therefore, it is important to introduce the state of the art industry practice into our software engineering education program through industry collaboration in teaching and learning.

The effective employment of this collaborative teaching and learning model has influenced student learning positively over a sustained period. The practical hands-on nature of the MeNtOR-based student project, the application of comprehensive system development methodology, the team work, the formal project presentations, and the critical analysis of their work all

make for challenging and stimulating learning experiences for the students. They have developed a set of practical and professional skills that are central to their future roles as industry professionals. Consequently their career prospects have been enhanced. As software engineering is a relatively immature engineering discipline professional practice is even more important than the other traditional engineering discipline [2]. We believe that industry involvement in software engineering education is an important force for promoting quality and competitiveness of software engineering graduates.

### References

- [1] IEEE Computer Society, *IEEE STD 610.12-1990, IEEE standard glossary of software engineering terminology*, 1990.
- [2] IEEE Computer Society and ACM, *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, 2004, Accessed at <http://sites.computer.org/ccse/SE2004Volume.pdf>.
- [3] Selic, B., What I Wish I Had Learned in School: Reflections on 30+ Years as a Software Developer. In *Proc. of 18th Conference on Software Engineering Education and Training (CSEE&T)*, 2005.
- [4] Teles, V. and Oliveira, C., Reviewing the curriculum of software engineering undergraduate courses to incorporate communication and interpersonal skills teaching. In *Proc. of the 16th Conference on Software Engineering Education and Training*, 158-165, 2003.
- [5] *National Association of Colleges and Employers, Job Outlook 2003*. Accessed at <http://www.naceweb.org/>.
- [6] Brooks, F. P., *The Mythical Man-Month, Essays on Software Engineering*, Anniversary Edition, Addison-Wesley, 2005.
- [7] Daniels, M., Faulkner, X., and Newman, I., Open ended group projects, motivating students and preparing them for the real world. In *Proc. of 15th Conference on Software Engineering Education and Training (CSEE&T)*, IEEE, 2002.
- [8] Alfonso, M. and Mora, F., Learning software engineering with group work. In *Proc. of 16th Conference on Software Engineering Education and Training*, 309 – 316, 2003.
- [9] Brereton, P., Lees, S., Gumbley, M., Boldyreff, C., Drummond, S., Layzell, P., Macaulay, L., Young, R., Distributed group working in software engineering education. *Information Software Technology*, **40**(4): 221-227, 1998.
- [10] Robillard, P., Measuring team activities in a process-oriented software engineering course. In *Proc. of 11th Conference on Software Engineering Education and Training*, 90-101, 1998.
- [11] Keen, C., Lockwood, C., and Lamp, J., A Client-focused, Team-of-Teams Approach to Software Development Projects. In *Proc. of Software Engineering Education & Practice*, 34-41, IEEE Computer Society Press, 1998.

# Data Flow Analysis and Testing for Web Service Compositions Based on WS-BPEL

Chien-Hung Liu  
*Dept. of Computer Science and  
Information Engineering  
National Taipei Univ. of Technology  
cliu@ntut.edu.tw*

Shu-Ling Chen  
*Dept. of Management and  
Information Technology  
Southern Taiwan University  
slchen@mail.stut.edu.tw*

## Abstract

*WS-BPEL is a business process execution language used to specify the behavior of a business process as compositions of Web services. As WS-BPEL is widely adopted as a standard for composing services, it becomes critical to analyze and test the WS-BPEL process in order to ensure the correctness of service compositions. However, most recent researches mainly focus on the development of WS-BPEL applications. There is little attention paid to WS-BPEL testing. This paper identifies and discusses various usages of data in WS-BPEL. A test model is proposed to capture the data flow artifacts of WS-BPEL with considerations of its specific features, such as concurrency and synchronization. Based on the model, data flow testing criteria can be used to select test paths for uncovering the data anomalies of the WS-BPEL process.*

## 1. Introduction

Web service composition has received much attention recently as it becomes an emerging approach for business to develop service-oriented applications. Among various service composition methods, Web Services Business Process Execution Language (WS-BPEL) [1] is considered a promising approach to specify the interactions of multiple Web services since it is advocated by a group of industry leaders, including IBM, Microsoft, and SAP, and has been standardized by OASIS [2].

WS-BPEL is a specification language used to describe business process behavior based on Web services. It allows users to specify the activities of a business process as Web services and to define logical flow of message exchanges among the Web services in order to accomplish the process tasks. The WS-BPEL process then can be executed to orchestrate the

corresponding Web services so as to support business transactions and process automation.

In particular, WS-BPEL is an XML-like programming language. It provides a rich set of constructs to specify a business process, including its process flow, data manipulation, exception handling, and compensation. A WS-BPEL process is composed of a set of interconnected basic and structured activities. The basic activities, such as receive, reply, invoke, and assign, mainly describe the elemental step of a business process. The structured activities, such as sequence, flow, if, and while, define the control flow of a process and contain other basic and/or structured activities recursively.

As WS-BPEL has emerged as an industry standard and is widely used for composing Web services, it is important to ensure that the WS-BPEL process is programmed correctly and the interactions among the Web services are handled properly. However, the complex XML descriptions and the WS-BPEL syntactic constructs can make the WS-BPEL process difficult to understand and test. Specifically, the semantics of WS-BPEL, such as concurrency and synchronization, pose new testing challenges and requires to be addressed in order to ensure the correctness of WS-BPEL process.

This paper aims to explore the potential of using data flow testing techniques to test the WS-BPEL process. Most specifically, we identify and analyze the possible data flow test artifacts introduced by WS-BPEL. A test model is proposed to abstract the data flow information of various WS-BPEL constructs with the considerations of the concurrency and synchronization semantics. Based on the proposed model, traditional data flow testing criteria can be used to select test paths for verifying if the data of the WS-BPEL process are handled properly.

The remainder of this paper is organized as follows. Section 2 briefly reviews recent work on testing WS-

BPEL process. Section 3 describes the data flow analysis of WS-BPEL. Section 4 presents a test model for representing the data flow artifacts of WS-BPEL process. Section 5 describes the uses of testing criteria to derive test paths for a WS-BPEL process based on the test model. Section 6 provides the conclusion remarks and our future work.

## 2. Related Work

Recently, there is an increasing interest in verifying and testing WS-BPEL process. Several existing methods, such as model checking, unit testing [3][4], and flow graph-based approaches, have been adapted to validate the WS-BPEL process. This section briefly reviews several WS-BPEL flow graph-based testing approaches that are related to our work.

Mei et al. [5] describe the possible data implications introduced by XPath Query in WS-BPEL applications. They illustrate that the data retrieved by XPath from XML messages can affect the execution workflow of the WS-BPEL process. To capture the data interactions between XPath and WS-BPEL, an X-WSBPEL model is proposed. The model annotates XPath expressions on the control flow graph of WS-BPEL. The structure of each annotated XPath expression is represented using an XPath Rewriting Graph (XRG) that describes the paths conceptually defined in the XPath expression. Based on the X-WSBPEL model, several testing criteria are proposed to derive test paths for exercising the data interactions between the XPath and WS-BPEL.

Bartolini et al. [6] address the issues regard testing Web service compositions (WSC) using data flow modeling approaches. They outline and classify several WSC data models which can be constructed from the requirement specification, BPEL implementation, or data property specification. The research issues about how these data models can be used individually or together to validate the data flow of WSC are discussed. A case study is also provided to demonstrate the WSC validation using the data flow models derived from the requirements and BPEL descriptions.

Hou et al. [7] present a data flow testing approach for Web service compositions. In their approach, the data defining and using information is extracted from the BPEL and WSDL documents. The data usage information is then annotated on the control flow graph of the BPEL process in which a mechanism is proposed to represent the situation of Dead Path Elimination (DPE). Based on the flow graph, an algorithm is presented to generate the test paths for detecting the data flow anomalies of the process.

Yuan et al. [8] present a BPEL Flow Graph (BFG) to represent the control flow of a BPEL process by extending traditional CFG to model the concurrency and synchronization semantics of BPEL specification. Based on the BFG, concurrent test paths for the BPEL process can be derived and their path conditions are computed using a matrix-based algorithm. The path conditions are then analyzed using a constraint solver in order to generate test data for each feasible path.

Yan et al. [9] describe an extended Control Flow Graph (XCFG) to model the execution flow of a BPEL process. Based on the XCFG, all the possible sequential flow paths are generated. These paths are then combined to form concurrent test paths. The constraints of the concurrent test paths are computed using a symbolic execution method. Finally, a BoNuS constraint solver is used to solve the path constraints and generate feasible test cases.

Endo et al. [10] propose a strategy for testing Web service compositions based on the structures of parallel programs. They present a Parallel Control Flow Graph (PCFG) to model the data flow and message exchanges between the parallel BPEL processes by taking into account the invoke, reply, receive, and pick activities. With the PCFG model, the intra-process and inter-process data usages are identified and a set of coverage criteria for parallel BPEL processes are provided to guide the test case selection and to analyze test coverage.

Lertphumpanya and Senivongse [11] describe a testing method for WS-BPEL processes based on the basis path testing technique. In their method, a traditional CFG is employed to represent the execution flow of WS-BPEL. From the CFG, basis paths for testing the WS-BPEL process are computed. A testing tool that supports the proposed method is also presented. However, they did not describe how to model the concurrency and synchronization using their CFG and how to generate basis paths in such cases.

Liu et al. [12] propose a BPEL Control Flow Graph (BCFG) to represent the control flow of WS-BPEL process. In particular, the BCFG adapts a subset of BPMN notations to model different constructs of WS-BPEL as well as their semantics including concurrency, synchronization, and dead path elimination. Based on the BCFG, an algorithm is presented to traverse the BCFG and to collect the path conditions for deriving feasible test paths.

## 3. Data Flow Analysis of WS-BPEL

Data flow testing is a structural testing technique for detecting improper uses of data. It focuses on how

a program variable is defined and used along the control flow of the program. A variable can be used in computation (c-use) or in predicate (p-use). Test paths of a program are selected based on testing criteria [13] and def-use chains (or def-use pairs) of variables, where a def-use chain is a path from definition to use without any other intervening redefinition [14].

To analyze the data flow of a WS-BPEL process, the characteristics of WS-BPEL need to be considered. Specifically, in the WS-BPEL process, variables can hold XML messages that represent a part of process state to be exchanged with partners. According to the message type defined in the WSDL document, a message can consist of one or more logical parts that hold the content of the message. A WS-BPEL process can access an entire message through the variable or extract a certain part of the message using XPath expressions [15]. For example, as shown in Figure 1, the variable *request* holds a message which consists of three parts: *firstName*, *name* and *amount*.

WS-BPEL document	WSDL document
<pre> ... &lt;variable&gt; &lt;variable name="request" messageType="creditMSG"/&gt; &lt;variable name="risk" messageType="riskMSG"/&gt; &lt;variable name="approval" messageType="approvalMSG"/&gt; &lt;/variable&gt; ... </pre>	<pre> ... &lt;wsdl:message name="creditMSG"&gt; &lt;wsdl:part name="firstName" type="string"/&gt; &lt;wsdl:part name="name" type="string"/&gt; &lt;wsdl:part name="amount" type="integer"/&gt; &lt;/wsdl:message&gt; &lt;wsdl:message name="riskMSG"&gt; &lt;wsdl:part name="level" type="string"/&gt; &lt;/wsdl:message&gt; ... </pre>

**Figure 1. An example of WS-BPEL and WSDL messages**

Depending on whether an entire message or a certain part of the message is defined or used, this paper denotes a variable as  $v$  or  $v.p$ , where  $v$  represents the variable name and  $v.p$  represents the part name  $p$  of the variable  $v$ . Moreover, to simplify the data flow analysis of WS-BPEL, when a part  $v.p$  is already defined, a use of the variable  $v$  is considered as a use of  $v.p$ . However, if a variable  $v$  is defined, a use of part  $p$  of  $v$  remains as a use of  $v.p$ .

In addition, variables in a WS-BPEL process can only be defined or used within the activities that exchange messages between the process partners, such as receive, invoke, reply, and pick, or within the activities that update data or control process flow, such as assign, if, and while. The definitions and uses of variables can vary in different WS-BPEL activities. For instance, when a receive activity is performed, the  $\langle$ variable $\rangle$  attribute of the activity (or its equivalent  $\langle$ fromPart $\rangle$  elements) will be defined with the inbound message. If a request-response type of invoke activity is executed, the data in the  $\langle$ outputVariable $\rangle$  attribute of the activity (or its equivalent  $\langle$ fromPart $\rangle$  elements)

will be used as a request message sent to corresponding partners. On the other hand, the  $\langle$ inputVariable $\rangle$  attribute of the invoke activity (or its equivalent  $\langle$ toPart $\rangle$  elements) will be defined with the response message.

Table 1 summarizes the activities those can be used to manipulate the WS-BPEL variables. To facilitate data analysis, the attributes and associated elements of each activity those involve data definitions and uses are also identified.

**Table 1 The definitions and uses of variables in the WS-BPEL activities**

Activity	definition	c-use	p-use
receive	variable attribute or $\langle$ fromPart $\rangle$ element		
reply		variable attribute or $\langle$ toPart $\rangle$ element	
invoke	outputVariable attribute or $\langle$ fromPart $\rangle$ element	inputVariable attribute or $\langle$ toPart $\rangle$ element	
assign	$\langle$ to $\rangle$ element	$\langle$ from $\rangle$ element	
wait			$\langle$ for $\rangle$ element $\langle$ until $\rangle$ element
if			$\langle$ condition $\rangle$ element
while			$\langle$ condition $\rangle$ element
repeatUntil			$\langle$ condition $\rangle$ element
forEach	counterName attribute	counterName attribute	counterName attribute
pick	variable attribute or $\langle$ fromPart $\rangle$ element		$\langle$ for $\rangle$ element or $\langle$ until $\rangle$ element
flow			$\langle$ joinCondition $\rangle$ element and $\langle$ transitionCondition $\rangle$ element

Moreover, WS-BPEL allows synchronization of concurrent activities. Thus, a variable can be defined or used by multiple concurrent activities. This can result in nondeterministic data flow behavior. For example, as shown in Figure 2(a), the same variable can be defined in two concurrent activities. This results in two possible def-use pairs that are exclusive of each other. In Figure 2(b), the variable can be defined in one concurrent activity and be used by the other. A def-use pair can exist between the concurrent activities only if the activity defining the variable is executed before the activity using the variable. The nondeterministic data flow may not appear if concurrent activities are synchronized using links. As shown in Figures 2(c) and 2(d), the concurrent activities are synchronized. In such case, the execution of parallel activities is restricted to be serial and data flow analysis for sequential programs can be employed.

Note that the WS-BPEL supports the mechanism of dead path elimination. If the join condition of a target activity evaluates to true, the activity will be executed. If, however, the join condition is false and the

<suppressJoinFailure> attribute of the activity is set to “yes,” the activity will be skipped and all outgoing links from the skipped activity will be set to false to avoid deadlocks. This indicates that a variable definition (or use) can be skipped when DPE occurs. For example, in Figure 2(e), depending on whether DPE occurs or not, the activity with definition  $def_2$  can be either executed or skipped. In such a case, we can derive two possible def-use pairs ( $def_1$ -use or  $def_2$ -use).

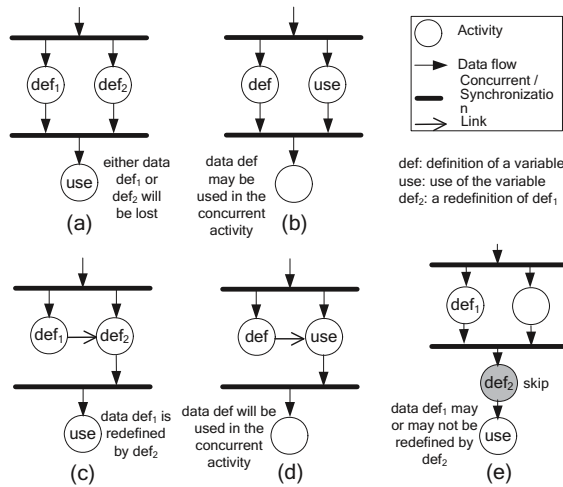


Figure 2. Examples of data definitions and uses in concurrent WS-BPEL activities

#### 4. WS-BPEL Data Flow Test Model

To facilitate data flow analysis, a test model is proposed to represent the data flow artifacts of WS-BPEL process. The test model is an extension of the WS-BPEL control flow graph (BCFG) in [12] by annotating with the data flow information. Figure 3 shows the graphical elements that are used to represent the BCFG constructs. Basically, a basic activity is represented using a normal node. A structured activity then can be constructed using normal, fork/join, and branch/merge nodes. The flow between the activities is indicated using sequence and conditional edges, where sequence edges represent execution flow and conditional edges represent flow paths with condition expressions, such as transition conditions of links.

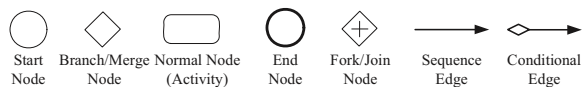


Figure 3. The graphical symbols of BCFG

Notice that, according to the semantic of DPE, when DPE is enabled and a target activity is not performed, the outgoing links of the target activity are

assigned a false value. This false value will be propagated along entire paths formed by successive links until a join condition is reached and is evaluated to true. To model this semantic statically in the BCFG, the branch condition for a target activity is defined as  $JC \wedge (JC_1' \vee JC_2' \dots \vee JC_n')$ , where  $JC$  is the join condition of the target activity and  $JC_i'$ ,  $1 \leq i \leq n$ , is the join condition of the source of the activity's incoming link as shown in Figure 4.

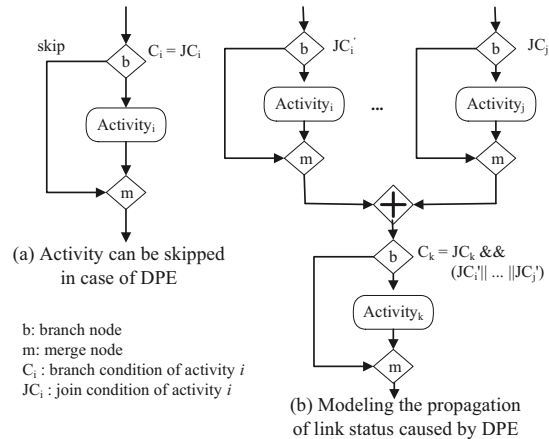


Figure 4. The static model of DPE in BCFG

To illustrate the proposed test model, we use a loan approval process that is described in the WS-BPEL specification [1]. Basically, the loan approval process receives a loan request from customer and generates either a “loan approved” or “loan rejected” message depending on the amount of request and the risk level associated to the customer. The process will invoke an assessor service to assess the risk associated with the customer and invoke an approver service to evaluate the loan request. For a low-risk customer with a requested amount less than \$10,000, the loan is approved automatically. Otherwise, the loan needs to be examined by an approver.

Figure 5 shows the BCFG annotated with data flow information for the loan approval process. In Figure 5, there are five normal nodes representing the basic activities in the process. These basic activities are all embedded in a flow activity that is constructed using normal, fork/join, and branch/merge nodes. In addition, all the basic activities are synchronized with links, where each link can have an associated transition condition and is represented by a conditional edge. Except for the receive activity, all other activities are targets of the links. Each target activity has a join condition that is the disjunction of link status of its incoming links. The join condition is a part of the branch condition for determining if the target activity can be skipped in case that DPE occurs.

To represent data flow information, the nodes and edges of the BCFG are annotated with corresponding definitions and uses of variables. For example, upon receiving the loan request message, the receive activity in node  $n1$  will assign a definition to the  $request$  variable (denoted as  $def(request, n1)$ ). This variable will be used in node  $n2$  as the request message sent to the assessor service partner (denoted as  $c-use(request, n2)$ ). Notice that a part of the  $request$  variable (i.e., amount) can also be used in the link transition conditions associated to edges  $e3$  and  $e8$  (i.e.,  $request.amount < 10000$  and  $request.amount \geq 10000$ ). However, a transition condition simply affects the link status which will be used in the join condition for determining whether the target activity of the link can be executed or an exception (or a DPE) will be raised. Thus, in the proposed test model, we will annotate the data flow artifacts of a transition condition in its associated join condition where two different flow branches can be introduced. As a result, the predicate use of the request amount in the transition condition of edge  $e3$  (or  $e8$ ) is annotated in the edges  $e4$  and  $e5$  (or edges  $e16$  and  $e17$ ).

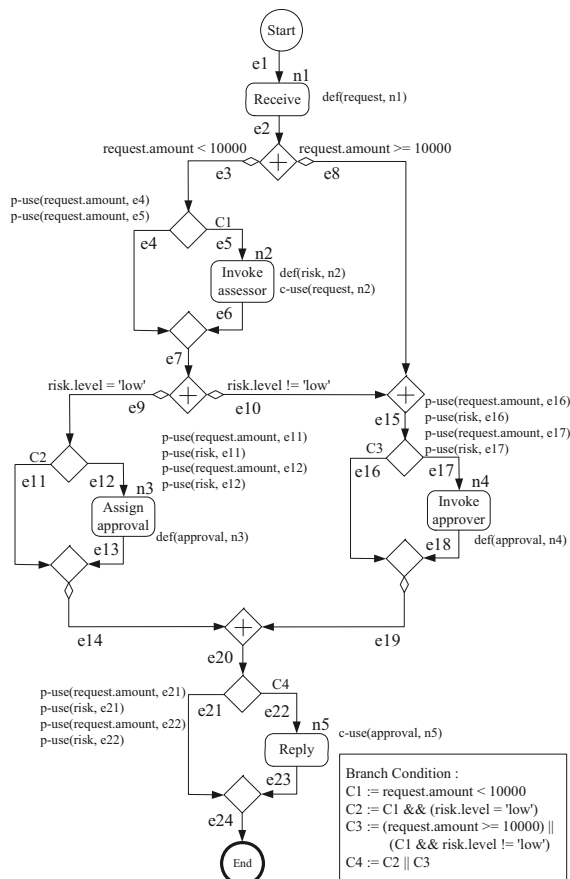


Figure 5. Test model of loan approval process

Table 2 shows the possible def-use chains for the loan approval process that are derived from Figure 5. Beware that the data flow artifacts of Web service compositions can be analyzed from the intra-process and inter-process perspectives. Table 2 shows only the intra-process def-use chains of the WS-BPEL process because the data flow information of partner Web services may not be available.

Table 2 The def-use chains of the loan approval process

Variables	Definition-Use Chains
$request$	(n1,n2)
$request.firstName$	(n1,n2)
$request.name$	(n1,n2)
$request.amount$	(n1,n2), (n1,e4), (n1,e5), (n1,e11), (n1,e12), (n1,e16), (n1,e17), (n1,e21), (n1,e22)
$risk$	(n2,e11), (n2,e12), (n2,e16), (n2,e17), (n2,e21), (n2,e22)
$approval$	(n4,n5), (n3,n5)

## 5. Selection of Test Paths

Based on the test model, test paths can be derived by traversing the BCFG. Traditional data flow testing criteria [13], such as all-du paths and all-uses, then can be applied to guide the selection of the derived test paths to uncover possible defects in data usage during the process execution.

Table 3 shows the possible test paths for the loan approval process that are selected based on the all-uses coverage criterion. The all-uses criterion requires that there exists at least one path from every definition of each variable to all possible uses of the variable that can be reached by the definition. In Table 3, each test path is represented as a sequence of edges in the BCFG, where the notations  $\cdot$  and  $\parallel$  indicate sequential and parallel flow, respectively.

Moreover, Table 3 also shows the path condition associated to each test path. The path condition can be used to decide if a path is feasible. A path is infeasible if there is a contradiction in its path condition. By examining the path condition of each path in Table 3, we can find that test path P4 is infeasible since P4 has a contradiction in its path condition  $C1 \wedge \neg C2 \wedge C3 \wedge \neg C4$ . This indicates that the def-use pairs (n1, e21) and (n2, e21) will not be covered by the selected paths.

## 6. Conclusions and Future Work

In this paper, we have proposed a data flow testing approach for Web service compositions based on WS-BPEL. The characteristics of WS-BPEL variables are described. WS-BPEL activities that can define and use the variables are identified. The features of

**Table 3 Test paths for the loan approval process**

No	Test Path	Path Condition	Covered Def-use Chain
P1	1-2-((3-5-6-7)  8)-((9-11-14)  ((10-15-17-18-19)))20-22-23-24	$C1 \wedge \neg C2 \wedge C3 \wedge C4$	(n1,n2), (n1,e5), (n1,e11), (n1,e17), (n1,e22), (n2,e11), (n2,e17), (n2,e22), (n4,n5)
P2	1-2-((3-5-6-7)  8)-((9-12-13-14)  ((10-15-16-19)))20-22-23-24	$C1 \wedge C2 \wedge \neg C3 \wedge C4$	(n1,n2), (n1,e5), (n1,e12), (n1,e16), (n1,e22), (n2,e12), (n2,e16), (n2,e22), (n3,n5)
P3	1-2-((3-4-7)  8)-((9-11-14)  ((10-15-17-18-19)))20-22-23-24	$\neg C1 \wedge \neg C2 \wedge C3 \wedge C4$	(n1,e4), (n1,e11), (n1,e17), (n1,e22), (n4,n5)
P4	1-2-((3-5-6-7)  8)-((9-11-14)  ((10-15-17-18-19)))20-21-24	$C1 \wedge \neg C2 \wedge C3 \wedge \neg C4$	(n1,n2), (n1,e5), (n1,e11), (n1,e17), (n1,e21), (n2,e11), (n2,e17), (n2,e21)

concurrency, synchronization, and DPE those complicate the data flow analysis of WS-BPEL are also discussed. A test model that can abstract the data flow information of WS-BPEL process is proposed. Based on the model, def-use chains for the variables of interest in the WS-BPEL process can be obtained. Traditional data flow testing criteria can be adapted to guide the selection of test paths for uncovering the data anomalies of the WS-BPEL process.

We are currently developing a data flow analysis algorithm for WS-BPEL and are building tools to support automatic construction of test model and generation of test paths. In the future, we plan to extend the test model to abstract the data flow artifacts of the fault handling and compensation mechanisms as well as other WS-BPEL specific features.

## 8. References

- [1] Web Services Business Process Execution Language Version 2.0, OASIS Standard, <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- [2] OASIS, <http://www.oasis-open.org/home/index.php>
- [3] Z. Li, W. Sun, Z.-B. Jiang, and X. Zhang, "BPEL4WS Unit Testing: Framework and Implementation," In *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*, 2005, pp. 103-110.
- [4] P. Mayer and D. Lübke, "Towards a BPEL Unit Testing Framework," In *Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications*, 2006, pp. 33- 42.
- [5] Lijun Mei, W.K. Chan, and T.H. Tse, "Data Flow Testing of Service-Oriented Workflow Applications," In *Proceedings of the 30<sup>th</sup> International Conference on Software Engineering (ICSE'08)*, May 2008, pp.371-380.
- [6] Cesare Bartolini, Antonia Bertolino, Eda Marchetti, and Ioannis Parissis, "Data Flow-Based Validation of Web Services Compositions: Perspectives and Examples," *Architecting Dependable Systems V*, LNCS, Vol. 5135, 2008, pp. 298-325.
- [7] Jun Hou, Baowen Xu, Lei Xu, Di Wang, and Junling Xu, "A Testing Method for Web services Composition Based on Data-Flow," *Wuhan University Journal of Natural Sciences, Springer-Verlag*, 2008, pp.40.
- [8] Yuan Yuan, Zhongjie Li, and Wei Sun, "A Graph-Search Based Approach to BPEL4WS Test Generation," In *Proceedings of the International Conference on Software Engineering Advances (ICSE'06)*, Oct. 2006, pp.14-14.
- [9] J. Yan, Z. Li, Y. Yuan, W. Sun, and J. Zhang, "BPEL4WS Unit Testing: Test Case Generation Using a Concurrent Path Analysis Approach," In *Proceedings of the 17<sup>th</sup> International Symposium on Software Reliability Engineering (ISSRE'06)*, Nov. 2006, pp.75-84.
- [10] A. T. Endo, A. S. Simão, S. R. S. Souza, and P. S. L. Souza, "Web Services Composition Testing: A Strategy Based on Structural Testing of Parallel Programs," In *Proceedings of Testing: Academic & Industrial Conference - Practice and Research Techniques (TAIC-PART'08)*, Aug. 2008, pp.3-12.
- [11] T. Lertphumpanya and T. Senivongse, "Basis Path Test Suite and Testing Process for WS-BPEL," *World Scientific and Engineering Academy and Society (WSEAS) Transactions on Computers*, Vol. 7, No. 5, May 2008, pp.483-496.
- [12] C.-H. Liu, S.-L. Chen, and X.-Y. Li, "A WS-BPEL Based Structural Testing Approach for Web Service Compositions," In *Proceedings of the 4<sup>th</sup> IEEE International Symposium on Service-Oriented System Engineering*, 2008, pp. 135-141.
- [13] P. G. Frankl and E. J. Weyuker, "An Applicable Family of Data Flow Testing Criteria," *IEEE Trans. Software Eng.*, Vol. 14. No. 10, 1988, pp. 1483-1498.
- [14] Rapps and E. J. Weyuker, "Selecting Software Test Data Using Data Flow Information," *IEEE Trans. Software Eng.*, Vol. SE-11, No. 4, 1985, pp. 367-375.
- [15] XML Path Language (XPath) 2.0, W3C, <http://www.w3.org/TR/xpath20/>



# Knowledge-based Software Test Generation

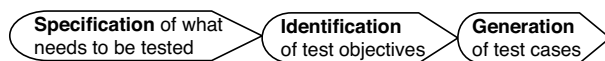
Valeh H. Nasser, Weichang Du, Dawn MacIsaac  
Faculty of Computer Science, University of New Brunswick  
Fredericton, NB, Canada  
{valeh.h, wdu, dmac}@unb.ca

## Abstract

*Enriching test oracles with a test expert's mental model of error prone aspects of software, and granting control to them to specify custom coverage criteria and arbitrary test cases, can potentially improve the quality of automatically generated test suites. This paper reports our investigation on the application of knowledge engineering techniques in automated software testing to increase the the control of test experts on test generation; ontologies and rules are used to specify what needs to be tested and reasoning is used for identification of test objectives, for which test cases are generated. An architecture of the ontology-based approach to testing is presented and a prototype which is implemented for unit testing is described with a case study.*

## 1. Introduction

Granting control to a test expert to utilize their knowledge about error-prone aspects of software and specify what needs to be tested, can result in generation of a smaller yet error-revealing test suite. At the current stage of the research we are focusing on model-based testing, but the proposed knowledge based approach can be generalized and applied to white-box and black-box software testing.



**Figure 1. Three Concerns of Test Generation**

There are three concerns in test generation (Figure 1): *specification* of what needs to be tested, *identification* of Test Objectives (TOs) based on the specification, and *generation* of test cases for the identified TOs. These three aspects can be tightly coupled.

The specification of what needs to be tested is addressed by definition of *test oracles* and *Coverage Criteria* (CC), which specify the correct behavior of the software and requirements on the generated test suite [21] respectively. This aspect of software testing is crucial, because it determines the quality of the generated test suite. Zhu *et al.* [21] categorize CC as *structural*, which specify what elements of software should be tested (such as All Transition Pair coverage [14]), *fault-based*, which uses some measurement of fault-detecting ability of the test suite (such as mutation-based methods [16]), and *error-based*, which is based on error-prone aspects of software (such as boundary testing [12]).

The second aspect of software testing is identification of test-objectives based on the specification of what needs to be tested. A test-objective delineates a single test case, and is identified with an algorithm. This concern can be tightly coupled with the first concern, because the identification algorithm can be tightly coupled with the specification of what needs to be tested. There are several approaches to identification of TOs: Explicit identification of TOs by a test expert [11]; the use of identification algorithms with the rules implicitly built into them [14]; provision of a language for defining CC rules and use of identification algorithms that rely on the specification language [8]; and translating CC into temporal logic and have a model checker to identify TOs [18].

The third concern in automated test generation is generation of test cases for the identified TOs. The test cases are generated based on a test oracle with several approaches: One approach is to use graph traversal algorithms [14, 4]. Another approach is using model-checking tools for test-case generation [19]. With this approach it is asserted that there is no path with the required specification in the model. The model checker tries to find the required path and returns it as an output. A third approach is using AI planners to generate test-cases [16]. AI Planners are used to generate paths

to reach identified goals.

Test oracles and CC are used for specification of what needs to be tested. The knowledge that is conveyed by test oracles can be at different levels of abstraction (i.e. code, design, or requirements [15]). An issue with abstraction is that poor abstraction can be a barrier to generating high quality test suites [5], if it removes the knowledge essential for specification of error-prone test cases. To solve this, Benz [5] demonstrates how abstraction of error-prone aspects of software can enhance test-case generation by defining system-specific CC. The error-prone aspects, which are also used in Risk Based Testing [1] are software elements that are more likely to produce an error and can be domain specific (such as concurrent methods, database replications, network connection), from general test guidelines and experience (such as boundary values), or system specific and revealed in interactions of testers with developers and designers (such as use of an unreliable library). Granting control to test experts to extend a test oracle with knowledge about error-prone aspects of the system and specify custom CC rules can increase the control of the test-expert on the automated test generation and, therefore, potentially enhance the quality of the generated test-suite.

To exploit a test expert's knowledge in automated test generation, the first two concerns of test case generation, i.e. specification of what needs to be tested and identification of TOs need to be decoupled. The decoupling makes the test oracle extensible and enables it to support specification of arbitrary test cases, implementation knowledge, invariants on model elements, distinguished states [17], and knowledge about error-prone aspect of the system. Also the system should support standard CC which are accepted in literature, as well as additional CC rules which are based on test experts' mental model.

In this work knowledge engineering techniques is used to decouple the specification of what needs to be tested and the identification algorithm of TOs and, therefore, enable a test expert to enrich the test oracle and specify custom CC accordingly. To achieve this, the test oracle and expert knowledge (EK) are represented in ontologies, which are connected together. The common CC rules and expert defined CC rules are used to specify what needs to be tested. Reasoning is used to identify the TOs. Then the test cases are generated for the identified TOs using model checking, AI planning, or graph traversal algorithms.

The rest of this paper is organized as follows. Section 2 describes how knowledge based approach is used to deal with the three aspects of software testing. Section 3 describes an implementation for unit testing

based on the UML State Machines (SMs). Section 4 illustrates a concrete example of unit test generation and specification of arbitrary expert knowledge. Section 5 concludes the paper.

## 2. Ontology based Approach

Our proposed ontology based approach to software test generation focuses on separation of the three concerns of test case generation as follows.

**Specification of What Needs To Be Tested** The knowledge that is used in specification of what needs to be tested is externalized in ontologies and rules. An extensible ontology based test oracle is connected to an expert's mental model. The mental model ontology is expert defined and can include knowledge about implementation, error-prone aspects, etc. Based on the vocabulary defined by the ontologies, CC rules, which are either standard or expert defined, are specified. CC rules are in the form shown below. The TO selection criteria specify a condition that should hold on some elements for them to be a part of structure of a test case.

TO :- TO selection criteria

The TOs are specified using the structural properties of corresponding test cases, which *directly* or *indirectly* make them candidates as TOs. For instance, in unit testing based on the UML SMs, a TO can specify that transition  $tr_1$  of the SM should be traversed immediately after transition  $tr_2$  is traversed. This TO can be directly required because every possible sequence of two transitions is required to be covered. This sequence can be indirectly required because the two transitions have a *definition-use* relationship [20], which is required to be tested.

**Identification of TOs** Reasoning on the test oracle and CC rules is used to identify test-objectives. Also, before a test case is generated for an identified TO, it is determined whether a test-case that satisfies the TO already exists in the test suite. This is done by reasoning on the partially-generated test suite, which is represented in an ontology. This approach reduces the number of redundant test cases. To this end, redundancy checking rules for a given TO need to be generated.

**Generation of Test Cases** A test-case for a given TO is generated using a test-case generation method such as AI planning, graph traversal, model checking,

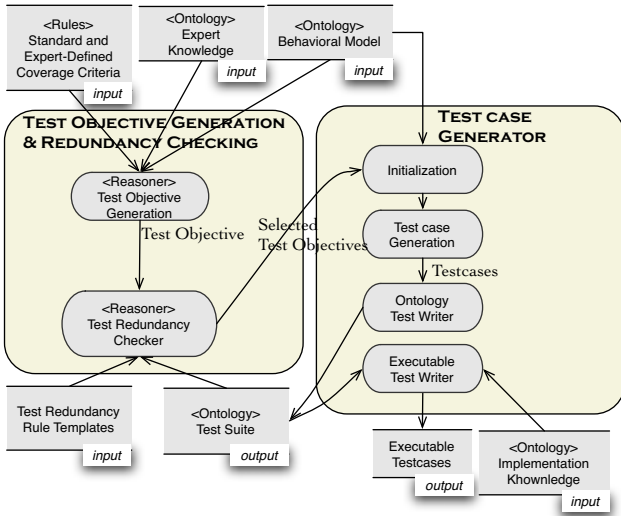


Figure 2. Ontology-based Test Generator

etc. and written in a programming language independent test-suite ontology. The executable test cases are then generated from the test suite ontology, using a programming language dependent implementation knowledge ontology. The implementation knowledge ontology, which can be reverse engineered from the source code and connected to the model ontology, conveys information such as name of implemented methods, state variable setters and getters, etc.

Figure 2 shows the architecture of the system which has two main processes: *TO Generation and Redundancy Checking* and *Test Case Generator*.

The TO Generation and Redundancy Checking is implemented using two reasoners. One reasoner is used to generate TOs from test oracle ontology which is connected to expert knowledge ontology, and CC rules. The other reasoner uses the redundancy checking rules and the test suite ontology which is generated so far to check whether a given test-objective is already satisfied in the test-suite or not. If it is not, the TO is accepted, otherwise it is discarded. If the TO is accepted, then a test case is generated for it and added to the test suite ontology. Then the redundancy checking reasoner continues to select another test-objective. The Test Case Generator is in charge of generating test cases for the selected TOs. The generated test cases are added to the test suite ontology, which is used to reduce redundant test-cases. The initialization subprocess, initializes the input of test case generation process. Finally the executable test cases are generated from the test suite ontology.

An ontology based representation of test oracle, and

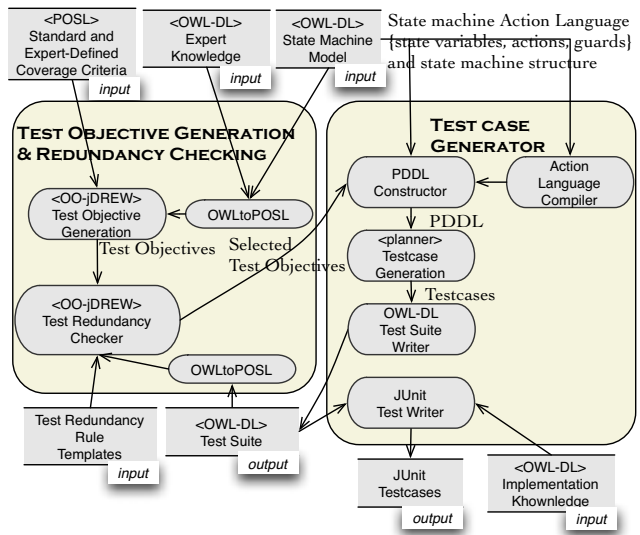


Figure 3. Implementation Technologies

rule based specification of CC, promotes separation of specification of what needs to be tested and identification of TOs, and, therefore, form a flexible mechanism for specification of various CC. The system empowers the test expert to use knowledge about the system to control the test-suite by specifying CC.

### 3. Implementation

The architecture that is discussed in the previous section is abstract and can be realized using various technologies. This section describes an implementation of the system for *SM based unit testing* and the technologies which are used to realize the system, including OWL-DL, POSL, and OO jDREW, and an AI planner named Metric-FF [10] (See Figure 3).

The UML SM, expert knowledge, implementation knowledge, and test suite are represented in OWL-DL [3]. A TBox ontology specifies different concepts and relations in them and an ABox ontology specifies a particular instances of them by importing the TBox ontology and instantiating the elements in it. The TBoxes are reusable while the A-Boxes are for an instance system. The Ontology Definition Metamodel (ODM), which is adopted by the OMG, has a section that describes the UML 2.0 metamodel in OWL-DL. However, a prototype ontology which is much simpler and less modifiable is used for the purpose of this work. The XMI [13] representation of the SM can be converted to the ontology-based representation automatically. The implementation knowledge can be automatically imported if the source code of the unit is available. Some

parts of the ontologies are visualized in next section.

The TO Generisation and Redundancy Checking component use OO jDREW [2] for reasoning tasks. The OWL-DL Ontologies are transformed into POSL [6] using the mappings presented by Grosz *et al.* [9]. The CC and the Test Redundancy Rule Templates are written and converted to POSL respectively. Examples of CC and Redundancy Checking Rule Templates are provided in the next section. The TOs are specified with *structure predicates* which specify some structural properties of the test case. Ideally a TO should specify why it is selected and its structure, to be used by the redundancy checking reasoner and the Test Case Generator process respectively. A set of structure predicates, which are used to specify the structural properties of the test cases are used to compose CC and TOs. Examples of TOs are provided in the next section.

The Test-case Generator process, uses an AI planner called Metric-FF [10] for test-case generation. The inputs to Metric-FF are the problem and domain description in the PDDL 2.1 language [7], which are provided by the planner initializer. The compiler-subprocess is in charge of parsing the state variables, guards and actions of the SM which are specified in an SM action language. The inputs of the planner is initialized based on data from the SM and structure predicates of a TO. The generated test cases, include methods to be called at each step, their inputs and the expected values of the state variables. The generated test cases are then given to the Test Suite Writer sub-process to be written back to the Test-Suite Ontology in OWL from which the JUnit test cases are generated.

#### 4. Case Study

Figure 4 visualizes the SM of a cross road traffic light controller. The traffic light stays green for at least ‘long time interval’ on one direction and turns yellow when a car is sensed in the other direction. Then it remains yellow for ‘short time interval’ before it becomes red. There is a correspondance between the SM elements and class under test: The state variables correspond to the state variables of the class; The events correspond to the methods; The actions simulate how the state variables are changed by the methods. The traffic light SM does not have a timer and delegates the counting responsibility to another class which produces call events.

Some parts of the ontological representation which is converted to POSL is visualized in Figure 5. A CC in POSL for All Transition Pair coverage and the query that is asked from OO jDREW to generate TOs are as follows:

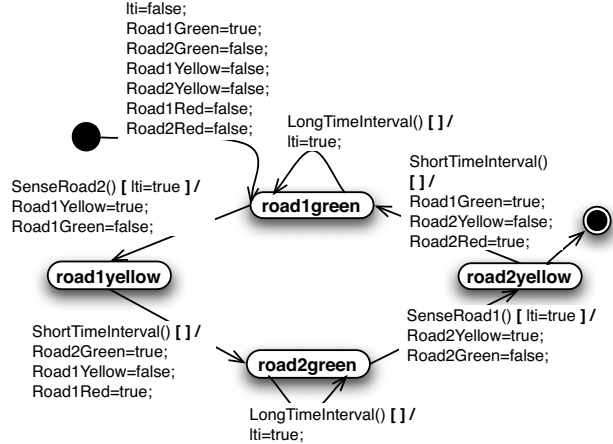


Figure 4. Traffic Light SM

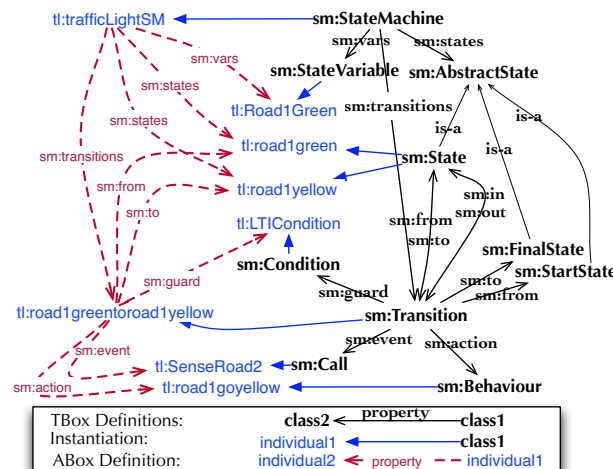


Figure 5. SM Ontology

```
Rule: coverage([immediate],[?a,?b]):-transition(?a),
transition(?b),notEqual(?a,?b),from(?a,?state),to(?b,?state).
Query: coverage (?predicates, ?args).
```

Before generating a test case for a TO, a redundancy checking rule is generated for it; test suite ontology (see Figure 6) is translated into POSL; and OO JDREW is used to check whether the TO is already satisfied. A TO and the corresponding redundancy checking rule are shown below:

```
coverage([immediate],[start2road1gr,road1grtoroad1gr]).
Redundancy Rule: exist(?t,?st1,?st2):-
hascall(?st1,start2road1gr),hascall(?st2,road1grtoroad1gr),
hasstep(?t,?st1),hasstep(?t,?st2),nextstep(?st1,?st2).
```

For an unsatisfied TO, AI planning is used to generate test cases which are added to test suite ontology ABox. To generate executable JUnit test cases, implementation knowledge ontology is used ( Figure 7).

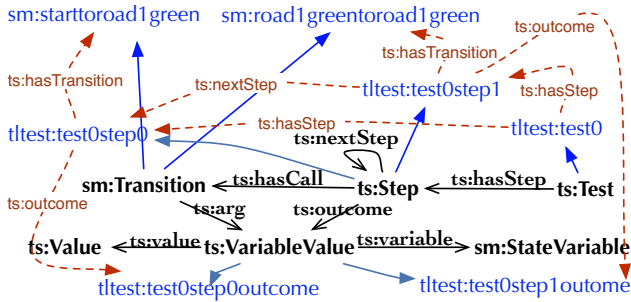


Figure 6. Test suite Ontology

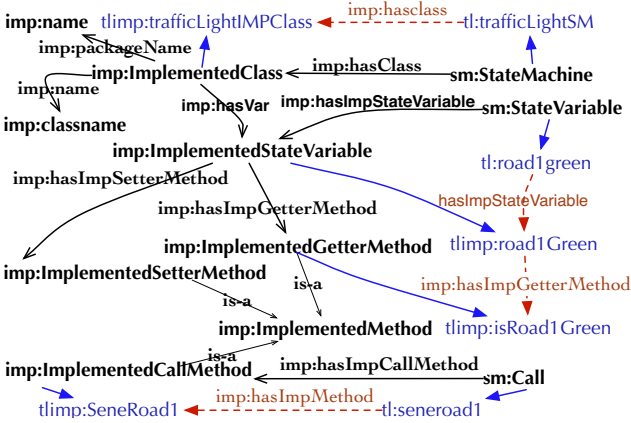


Figure 7. Implementation Knowledge

#### 4.1. Other Coverage Criteria

While some CC such as All Transition Pair coverage merely depend on the structure of a SM, some other CC refer to additional expert knowledge which can be expressed in an ontology and be used to define custom CC. As an example, a test expert's knowledge about use of an unreliable library can be modeled, and CC for a test suite that tests every call to the library once can be specified. This knowledge can be attached to the unit's SM ontology (Figure 8). A CC can be defined as it is shown below. The test structure indicated by this CC is that at a transition  $t$ , the state variable  $sv$ , must have value  $val$ .

```
coverage([AtTransitionStateVariableHasValue], [?t,?sv,?val]):-
  untestedlib(?l),method(?m), belongsto(?m,?l),behaviour(?b),
  uses(?b,?m),transition(?t),action(?t,?a),
  variablevalue(?vv), risky (?m,?vv), value(?val),
  statevariable (?sv),hasvar(?vv,?sv), hasvalue(?vv,?val).
```

Another example is a CC that defines if a state variable has a boundary value in a state, then a transition that has a behavior that uses the value should be tested

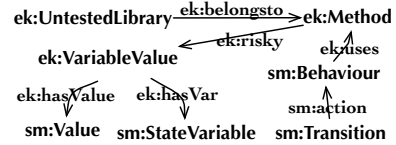


Figure 8. Use of an Unreliable Library

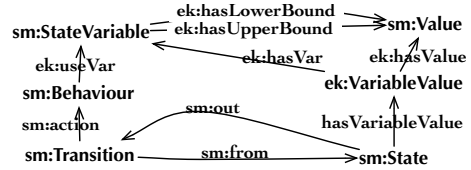


Figure 9. Boundary Values

[12]. Figure 9 visualizes the expert knowledge and the CC can be as follows:

```
coverage([AtTransitionStateVariableHasValue], [?t,?sv,?val]) :-
  state(?s), variablevalue(?vv), hasvar(?vv,?sv),
  hasvalue(?vv,?val), hasboundary(?sv,?val),transition (?t),
  from (?t,?s),behaviour(?b), action(?t,?b), usevariable(?b,?sv).
  hasboundary(?sv,?val):-lowerbound(?sv,?val).
  hasboundary(?sv,?val):-upperbound(?sv,?val).
```

Other CC can be implemented by representing the knowledge in an ontology and defining rules that refer to vocabulary defined by the ontology. For instance, to implement all content dependance relationship coverage [20], which requires that every use of a variable be sequenced after every definition of the variable in a test case, the definition-use relationships among the behaviors and guards can be added to the ontology. Another example is Faulty Transition Pair coverage [4], which required error states, the transitions to them be modeled in an ontology. Then a rule that generate objectives to go to error states are generated. Implementing Full Predicate coverage[14] required more effort.

## 5. Concluding Remarks

This work studies the use of knowledge engineering techniques in software testing. The benefit that knowledge engineering techniques can bring to automated testing is specification of extensible test oracles which can model test experts' mental model and lend themselves to definition of custom CC. In unit testing, a unit can be large and central to the system. The method grants control to a test expert to specify what test cases should be generated and therefore potentially increases a test suite's quality. Other benefits of the system are that the generated test suite ontology is programming

language independent and can be translated into different languages, and the expert knowledge TBox ontology is re-usable.

Providing for enriching the test oracle with additional knowledge, enables blending white-box and black box testing into gray box testing. While rigid test oracles and CC hinder a test expert in using their knowledge to control automated testing, an extensible CC allows them to add any knowledge to the test oracle and define CC accordingly. Either arbitrary test cases can be defined or rules can be used to generate TOs. Rules can be defined based on a model's structural elements and/or generally accepted, domain-specific, and/or system specific error prone aspects of software.

Although the system is extensible, manipulating the CC and knowledge required knowledge engineering skills. Further research into test experts' mental model, and a method of presenting the system to a test expert abstractly, so that it can be easily learnt is required. Also, the CC which are accepted by literature should be implemented in the system. Further research needs to be done into how test experts describe test cases and a language of structure predicates needs to be devised accordingly, for the test expert to specify the structure of test cases using TOs. This work concentrated on unit testing and the next step is using knowledge engineering in integration testing and system testing.

## References

- [1] J. Bach. Risk-based Testing. *Software Testing and Quality Engineering Magazine*, 1(6), 1999.
- [2] M. Ball. OO jDREW: Design and Implementation of a Reasoning Engine for the Semantic Web. Technical report, Technical report, Faculty of Computer Science, University of New Brunswick, 2005.
- [3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, et al. OWL Web Ontology Language Reference. *W3C Recommendation*, 10, 2004.
- [4] F. Belli and A. Hollmann. Test generation and minimization with "basic" statecharts. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 718–723, 2008.
- [5] S. Benz. Combining test case generation for component and integration testing. In *Proceedings of the 3rd international workshop on Advances in model-based testing*, pages 23–33, 2007.
- [6] H. Boley. POSL: An Integrated Positional-Slotted Language for Semantic Web Knowledge. <http://www.ruleml.org/submission/ruleml-shortation.html>, 2004.
- [7] M. Fox and D. Long. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20(2003):61–124, 2003.
- [8] G. Friedman, A. Hartman, K. Nagin, and T. Shiran. Projected state machine coverage for software testing. In *Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis*, pages 134–143, 2002.
- [9] B. N. Grosz, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 48–57, 2003.
- [10] J. Homann. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.
- [11] A. Howe, A. Mayrhauser, and R. Mraz. Test Case Generation as an AI Planning Problem. *Automated Software Engineering*, 4(1):77–106, 1997.
- [12] N. Kosmatov, B. Legeard, F. Peureux, and M. Utting. Boundary Coverage Criteria for Test Generation from Formal Models. In *Proceedings of the 15th International Symposium on Software Reliability Engineering*, pages 139–150, 2004.
- [13] Object Management Group. XML Metadata Interchange (XMI) specification. <http://www.omg.org/technology/documents/formal/xmi.htm>, 2007.
- [14] J. Offutt and A. Abdurazik. Generating tests from UML specifications. In *UML'99 - The Unified Modeling Language. Beyond the Standard*. Springer, 1999.
- [15] T. Ostrand and M. Balcer. The category-partition method for specifying and generating functional tests. *Communications of the ACM*, 31(6):676–686, 1988.
- [16] A. Paradkar. Plannable Test Selection Criteria for FSMs Extracted From Operational Specifications. In *Proceedings of the 15th International Symposium on Software Reliability Engineering*, pages 173–184, 2004.
- [17] A. Paradkar. A quest for appropriate software fault models: Case studies on fault detection effectiveness of model-based test generation techniques. *Information and Software Technology*, 48(10):949–959, 2006.
- [18] S. Rayadurgam and M. Heimdahl. Coverage based test-case generation using model checkers. In *Engineering of Computer Based Systems, 2001. ECBS 2001. Proceedings. Eighth Annual IEEE International Conference and Workshop on the*, pages 83–91, 2001.
- [19] S. Rayadurgam and M. Heimdahl. Coverage Based Test-Case Generation using Model Checkers. In *Eighth IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, page 83, 2001.
- [20] Y. Wu, M. Chen, and J. Offutt. UML-Based Integration Testing for Component-Based Software. In *Cots-Based Software Systems: Second International Conference*, pages 251–260, 2003.
- [21] H. Zhu, P. Hall, and J. May. Software unit test coverage and adequacy. *ACM Computing Surveys (CSUR)*, 29(4):366–427, 1997.

# SOME EXPERIMENTS ON TEST CASE TRACEABILITY

Macario Polo, Beatriz Pérez and Pedro Reales  
Department of Information Systems and Technologies  
University of Castilla-La Mancha  
Paseo de la Universidad, 4  
13071-Ciudad Real (Spain)  
{macario.polo,beatriz.plamanca,pedro.reales}@uclm.es

## Abstract.

*This paper investigates the relationships between the test cases generated from state machines and their ability to find faults in the implementation of the class proceeding from the state machine. Many research works have proposed strategies to generate test cases for different kinds of software artefacts. To our best knowledge, the “traceability” of the coverage got by the test cases in different abstraction levels has not been studied, so being this work the first (or one of the first) contribution on this sense.*

## 1. INTRODUCTION

Testing is a process which involves all the activities of the software life cycle. Different authors have proposed techniques for preparing test cases from the early stages of software development [1-5], some of which are reviewed in the next section of this paper.

Basanieri et al. [1], for example, derive test cases for sequence diagrams corresponding scenarios: they translate the sequence of messages to test templates, which may be later combined with actual test data to get executable test cases. Each test template starts with messages with no predecessor (i.e., launched by actors) whose execution is continued by the dependent messages. Guards may imply the introduction of alternative messages. The test templates are combined with actual data to get test cases which likely may be executed against the actual system.

Other authors have described techniques and coverage criteria to get test cases from other types of diagrams: thus, the coverage criteria of Andrews et al. [5] (for UML class and interaction diagrams), could be used to check the quality of the test cases produced with the Basanieri et al.’s method.

These types of techniques emphasize the importance of testing from the beginning of software projects, so being close to the idea of continuous verification and validation, a broad concept which includes “testing”. In the two specific techniques so briefly summarized here, they also help to prepare the source code of tests which may be executed when a version of the system is available.

However, it is not clear the relationship between the quality of a test suite defined for a UML artefact and the quality reached by that very same suite on the executable code corresponding to that original artefact.

This paper presents a first contribution on this sense, with the analysis of the quality that test cases generated from state machines get on source code. The idea is repre-

sented in Figure 1: given the behaviour of a class, specified by means of a state machine, and a state-machine criterion-adequate test suite got by the application of some technique, the goal is to study whether the test suite is still adequate when an executable specification of the class is available: this is, does remain the coverage traceable when the abstraction level of a software artefact is decreased?

Since this work is focused on statecharts, a revision of some works on state machines and test case generation from state machines is presented in Section 2. Then, Section 3 describes two case studies and discusses the results obtained. Finally, Section 4 draws our conclusions and future lines of work.

## 2. RELATED WORK

This work concern is the analysis of the test correspondence between models and code in the context of “behavioural state machines”, as defined in the current UML specification [6]. For this context, Figure 1 illustrates the *Has a behaviour defined in* relationship, which corresponds to the relationship between the Class and State-Machine UML classifiers [7].

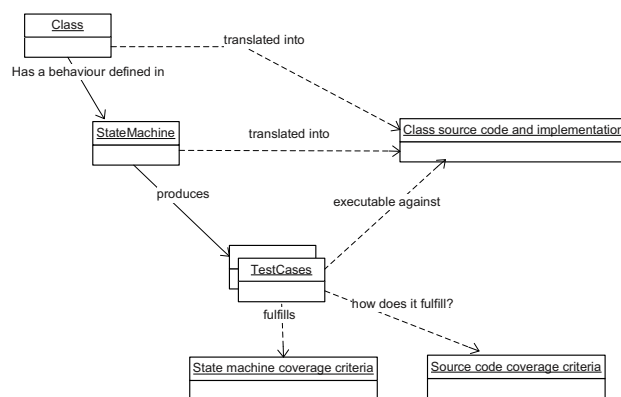


Figure 1. Schema of this proposal

The figure also shows the *translation* relationship between the specification of a class (whose behaviour is annotated with a state machine) and its implementation. Related to this translation, Warmer and Kleppe [8] informally describe an algorithm to translate class and state machines to source code.

According to their rules, the source state can be considered part of the precondition of the implementation of the transitions triggered by operation *calls*, whilst the target state could be part of the postcondition. If the transi-

tion has also a guard condition, this is considered part of the precondition. If the same event occurs for more than one transition, all possible situations must be taken into account in order to correctly write the preconditions, because it must be checked that the class instance is in an accepted state for triggering that transition.

Since actually a class state can be described as a function of the class fields' values, it is easy to get a source code specification from a state machine. If the mechanism to know the instance state is costly, Warmer and Kleppe propose to add a field representing the state to the class, which must be recalculated when any of the influencing fields changes, and which is queried before triggering the transitions.

Figure 2 shows a simplified algorithm, inspired in [8], to get the code from a class state machine: for each transition, it finds the corresponding method in the class description and builds a new *block of sentences* (for composing the precondition and the possible effect) which is added to the method. In a previous work related with a tool for database reverse engineering and code generation [9], we successfully used this very same algorithm to generate code for classes which proceed from database tables: the classes and their relationships are inferred from tables and foreign key relationships, and the software engineer may modify the default behaviour (provided by the tool) of the classes with the addition of one state machine to each class. The figure also includes the structure of both a Class as a StateMachine, which is also compatible with Figure 1.

```

Class=(Name, Fields, Operations, StateMachine)
operation=(Name, Parameters, Return, BlocksOfSentences)
StateMachine=(States, Transitions; Initial, End∈ States)
t ∈ Transitions=(Call, Guard, Effect)

function getCodeFromSM(class : Class) {
  ∀ t ∈ class.StateMachine.transitions {
    op=class.findOperation(t.call)
    Block b=new Block()
    b.addPrecondition(t.sourceState)
    b.addPrecondition(t.guardCondition)
    b.addBody(t.effect)
    op.add(b)
  }
}

```

**Figure 2. An algorithm for generating code from a state machine**

In summary, it is clear that: (1) the behaviour of a class can be formally described by means of a state machine; and (2) there is an effective means to generate the code corresponding to a class annotated with a state machine.

In order to know whether a test suite is adequate for testing a software artefact, the definition of one or more coverage criteria for such type of artefact is required. In this sense, Hong et al. [3] and Offutt et al. [10] have proposed the following coverage criteria for state machines:

- State coverage [3]. A test suite  $T$  satisfies *state coverage* if each state is covered by one or more test sequences in  $T$ .

- Transition [3, 10]. A test suite  $T$  satisfies this criterion if each transition is traversed by one or more test sequences in  $T$ .
- Full predicate [10]. For each predicate  $P$  on each transition and each test clause  $c_i$  in  $P$ ,  $T$  must include tests that cause each clause  $c_i$  in  $P$  to determine the value of  $P$ , where  $c_i$  has both the values *true* and *false*. A predicate is a boolean expression whose value may determine the triggering of a transition.
- Transition pair [10]. For each pair of adjacent transitions  $S_i : S_j$  and  $S_j : S_k$ ,  $T$  must contain a test that traverses each transition of the pair in sequence.
- Configuration [3]. According to these authors, a configuration is “maximal set of states in which a system can be simultaneously”. Thus, a test suite  $T$  satisfies this criterion if each configuration is covered by one or more test cases in  $T$ .
- Complete sequence [10].  $T$  must contain tests that traverse meaningful sequences of transitions on the state machine. “Meaningful sequences” are chosen by the test engineer based on experience, domain knowledge and other human-based knowledge.

Once a coverage criterion has been selected, the following step is the generation of test cases by the application of some algorithm which, in general, may be adopted from graph literature.

### 3. ANALYSIS OF TEST CASES TRACEABILITY

With “traceability”, we reference the degree that coverage reached in a software artefact described at a given, high abstraction level, is preserved when such artefact is translated into another at a lower level. This work is focused on the traceability of test cases from state machines to executable specifications, for which we have used two experiments, one extracted from literature and one which has been elaborated by us.

- 1) The first is the Cruise Control system, which has been used, among others, in references [10, 11], where the corresponding state machine can be found.
- 2) The second one corresponds to a Manager that controls the light flow of two semaphores, as a problem to be resolved by our students (Figure 3). When there are no pedestrians, the manager changes the light of both semaphores ( $a$  and  $b$ ) sending them the *change* event every a fixed number of seconds (60, 63, 66, 83, and 86). However, a pedestrian may request the red light in any of the semaphores: if the semaphore where red is requested is in yellow or red, nothing happens; if it is in green and the semaphore is  $a$ , then the managers changes  $a$  to yellow either 20 seconds after the request or, if less than 20 seconds remain, in this time. If the red light is requested on  $b$ , then the request is passed to  $a$ .

For both cases we have applied State, Transition and Transition pair coverage. Then: (1) the two state machines



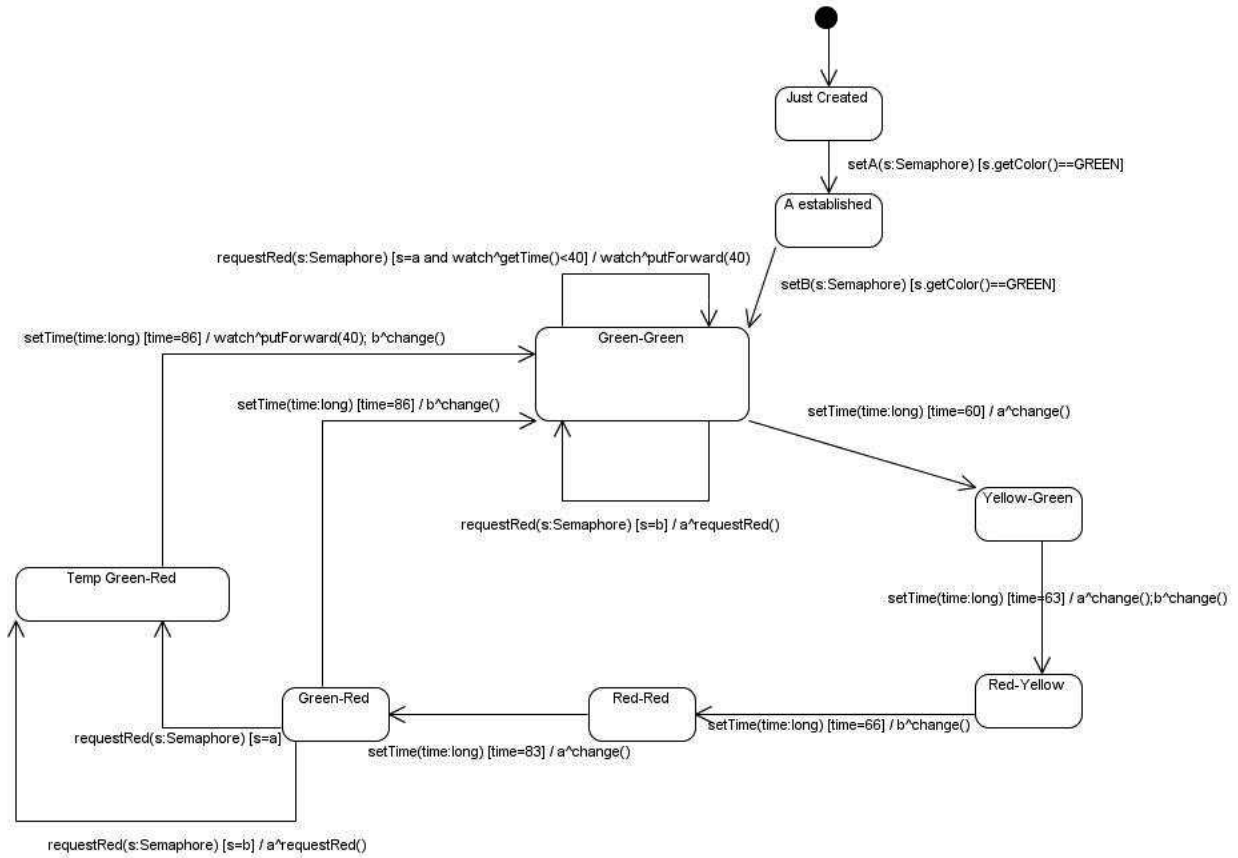


Figure 3. State machine for the semaphore's manager example

have been translated into Java programs, according to the afore-mentioned translation rules [8], and the test cases for the three coverage criteria were generated; (2) the test cases were then translated into executable Java test cases in MuJava and testooj [12, 13] formats (which are two tools for mutation testing); (3) the coverage reached by the Java test cases has been measured in terms of the mutation score using testooj [12].

### 3.1. Getting the source code

The state machines have been faithfully translated into Java programs according to the algorithm presented in Figure 2. As an example, Figure 4 shows the source code of the *requestRed* method in the semaphores' manager, which is one of the operations accepted by the Manager class (according to its state machine, Figure 3): each transition labelled with *requestRed* is collected with a precondition, which proceeds from the corresponding source state and the possible guard condition.

### 3.2. Getting the test cases

State and transition test cases (Figure 5) have been manually written, whilst transition pair cases have been generated with a simple program that goes through the state machines with that kind of route. This very same program translates them into JUnit and testooj test cases.

### 3.3. Quality of the executable test cases

To check the ability of the test suite to find faults, we have generated mutants with MuJava. To minimize the

cost of testing, *testooj* was used to reduce the mutant suites by applying second-order mutation [14]. Table 1 shows the number of second-order mutants generated and the results of the execution of the tests against the mutants (as the mutation score). In all the cases, the results obtained are far from the optimal score, which is 100%.

```

public void requestRed(Semaphore semaphore) {
    if (a.getColor()==GREEN && b.getColor()==GREEN) {
        if (semaphore==a) {
            long currentMiliseconds=this.watch.getCurrentMiliseconds();
            if (currentMiliseconds<40000)
                this.watch.putForward(40000);
        } else
            this.a.requestRed();
    } else if (a.getColor()==GREEN && b.getColor()==RED) {
        if (semaphore==a) {
            this.temporalState=true;
        } else
            this.a.requestRed();
    }
}
  
```

Figure 4. Source code of *Manager::requestRed*

Class	2 <sup>nd</sup> order mutants	Mutation score		
		State	Transition	Transition pair
CruiseControl	56	35%	46%	83%
Manager	47	61%	63%	80%

Table 1. Results obtained in the two experiments

### 3.4. Analysis of results

State coverage requires that each state in the machine is traversed by a test case. This implies that just one of the

transitions arriving the state is executed, maybe existing transitions corresponding to other methods that are not executed. Therefore, in terms of source code coverage, state coverage does not even imply method coverage.

Regarding Transition coverage, it means that each transition is triggered at least once, what usually implies the execution of each method of the corresponding class. Thus, coverage of transitions implies coverage of methods

The best results of Table 1 correspond to Transition pair coverage. According to the algorithm shown in Figure 2, when a transition pair  $(p,q)$  is traversed by a test case on the state machine, this means that the preconditions for  $p$ ,  $q$  (*source states + guard conditions*) have been fulfilled. This leads to that the executable test case forces the execution of the corresponding *true branch* in the conditional instruction (in the semaphores' manager, for example, a test case goes from Green-Green to Red-Yellow, leaving Yellow-Green with  $time=63$ ). The test case corresponding to the *false branch* is not generated if the state machine does not contemplate it (in the same example, the manager is not tested when it is in Yellow-Green and the *setTime* method is triggered with  $time \neq 63$ ). Thus, there may be test situations which, being almost completely tested on the state machine, are not tested enough on its corresponding implementation. This criterion implies partial All decisions coverage (All decisions is got when each decision in the source code is executed with *true* and *false*).

<p><b>Pair transition test cases for CruiseControl (ign=ignition; activ=activate; brk=brake; dct=dct; res=resume; tF=tooFast)</b></p> <p>[new, ign, ign, ign, act, tF, ign -]          [new, ign, ign, ign, act, tF, act, brk, res, tF -]          [new, ign, ign, ign, act, tF, act, brk, res, brk, act, tF -]          [new, ign, ign, ign, act, tF, act, brk, res, brk, act, brk, res, tF -]          ...</p> <p><b>Pair transition test cases for the semaphores' Manager (sT=setTime; rR=requestRed; watch.gT=watch.getTime; GR=GREEN)</b></p> <p>[new, setA(a) [s.getColor()==GR], setB(b) [s.getColor()==GR], sT(60), sT(63), sT(66), sT(83), sT(86), sT(60)]          [new, setA(a) [s.getColor()==GR], setB(b) [s.getColor()==GR], sT(60), sT(63), sT(66), sT(83), sT(86), rR(a) [watch.gT(&lt;40000), sT(60)]          [new, setA(a) [s.getColor()==GR], setB(b) [s.getColor()==GR], sT(60), sT(63), sT(66), sT(83), sT(86), rR(a) [watch.gT(&lt;40000), rR(a) [watch.gT(&lt;40000), rR(b), sT(60)]          [new, setA(a) [s.getColor()==GR], setB(b) [s.getColor()==GR], sT(60), sT(63), sT(66), sT(83), sT(86), rR(a) [watch.gT(&lt;40000), rR(a)          ...</p>
---

Figure 5. Some transition pair test cases

#### 4. CONCLUSIONS AND FUTURE WORK

This paper has presented an initial study about the keeping of test cases coverage between different representations of the same system, expressed at different abstraction levels. It has focused on thee coverage criteria for class state machines and their corresponding source code, and has been applied to two single case studies. Obviously, more experimentation with other systems is required. Ideally, these experiments should be carried out with benchmark systems, that allow to other researchers the replication of the experiments and the advance in this research line, that we believe quite interesting to a better understanding of

the need of applying validation and testing techniques along the whole life cycle.

#### 5. ACKNOWLEDGMENTS

This work is partially supported by the PRALÍN (Pruebas en Líneas de Producto) project, financed by the Government of Castilla-La Mancha and the European Social Fund, by the grant number PAC08-0121-1374.

#### 6. REFERENCES

1. Basanieri F, Bertolino A and Marchetti E (2002). *The Cow\_Suite Approach to Planning and Deriving Test Suites in UML Projects*. 5th International Conference on The Unified Modeling Language: Springer-Verlag. LNCS.
2. Baudry B, Traon YL and Sunyé G (2002). *Testability Analysis of a UML Class Diagram*. 8th IEEE Symposium on Software Metrics.
3. Hong HS, Lee I and Sokolsky O (2001). *Automatic test generation from statecharts using model checking*. Formal Approaches to Testing of Software (FATES'01). Aalborg, Denmark: BRICS: Basic Research in Computer Science.
4. Tse T and Xu Z (1996). *Test Case Generation for Class-Level Object-Oriented Testing*. 9th International Software Quality Week. San Francisco, CA.
5. Andrews A, France R and Ghosh S (2003). *Test adequacy criteria for UML design models*. Software Testing, Verification and Reliability, (13), p. 95-127.
6. OMG (2005). Unified Modeling Language: Superstructure version 2.0. Object Management Group.
7. OMG (2005). UML 2.0 OCL Specification. Object Management Group.
8. Warmer J and Kleppe A (2003). *The Object Constraint Language, 2nd edition*: Addison Wesley.
9. Polo M, García-Rodríguez I and Piattini M (2007). *An MDA-based approach for database reengineering*. Journal of Software Maintenance & Evolution: Research and Practice, 19(6), p. 383-417.
10. Offutt AJ, Liu S, Abdurazik A and Amman P (2003). *Generating test data from state-based specifications*. Software Testing, Verification and Reliability, (13), p. 25-53.
11. Offutt A (1999). *Generating test data from requirements/specifications: Phase II final report*. Technical Report ISE-TR-99-01, Department of Information and Software Engineering, George Mason University. Fairfax, VA.
12. Polo M, Piattini M and Tendero S (2007). *Integrating techniques and tools for testing automation*. Software Testing, Verification and Reliability, 17(1), p. 3-39.
13. Ma Y-S, Offutt J and Kwon YR (2005). *MuJava: an automated class mutation system*. Software Testing, Verification and Reliability, 15(2), p. 97-133.
14. Polo M, Piattini M and García-Rodríguez I (2008). *Decreasing the cost of mutation testing with second-order mutants*. Software Testing, Verification and Reliability, In press (DOI 10.1002/stvr.392).

# Business Modeling for Service Engineering: Toward an Integrated Procedure Model

Gregor Scheithauer, Stefan Augustin

Siemens AG - Corporate Technology

Information & Communication - Knowledge Management

Otto-Hahn-Ring 6, 81739 Munich, Germany

Email: {gregor.scheithauer.ext—stefan.augustin}@siemens.com

Guido Wirtz

University of Bamberg

Distributed and Mobile Systems Group

Feldkirchenstraße 21, 96052 Bamberg, Germany

Email: guido.wirtz@uni-bamberg.de

**Abstract**—Business modeling for service engineering aims at flexible transformation of business logic into software code. The ISE framework is an interdisciplinary approach which embraces this concept to engineer services. This paper discusses work related to business modeling, introduces the ISE framework, and examines three areas for improvements. The first improvement addresses a firm terminology in that it reduces term ambiguity. The second improvement proposes additional concepts and meta models to advance the framework's semantics. The last improvement presents an initial integrated procedure model with eleven steps, which will guide and support the modeler through the service engineering process.

**Keywords:** service engineering, procedure model, modeling

## I. INTRODUCTION

Business modeling is a discipline which depicts the relationship between business logic and its realization with information technology. The idea is to define business logic in a technology-free fashion and frictionless transform it into technical blueprints, neglecting traditional expensive and interminable software engineering projects, and hence, it eases the alignment between business requirements and IT. Recently, business modeling gained momentum in the domain of business process automation [20] and service engineering [8], since globalization and technological change [16] provoke highly dynamic environments as well as high uncertainties [15]. As a consequence, organizations need to adapt quickly and frequently.

In general, business logic is sub-divided into a strategic and a conceptual layer [20]. The strategic layer describes *what* needs to be done, whereas the conceptual layer states *how* this is accomplished [5] while still ignoring information technology which is only considered afterwards.

However, while much efforts were made to explore the relationship between the conceptual layer and information technology, few approaches exist which target the relationship between strategic aspects and their conceptualizations [20].

Kett et al. [8] took these new developments into consideration and developed the Inter-enterprise Service Engineering (ISE) framework in order to address these new challenges and to embrace the concept of business modeling.

This work's contribution is an enhancement of the ISE framework [8] in that it provides a finely granulated semantics

for both, the strategic and the conceptual perspective. Additionally, it proposes an initial procedure model to integrate the various aspects of services, and to guide the service engineering process.

The remainder of this paper is structured as follows: section II reviews related work and section III introduces the ISE framework. Section IV presents available concepts for improvement and discusses the final framework, whereas section V introduces the initial integrated procedure model. Section VI concludes this work and offers prospects about future work.

## II. RELATED WORK

Prior to diving into the ISE framework and the procedure model, this section discusses available work in the area of business modeling.

Bergholtz et al. [2] claim that two types of models exist in the e-commerce domain: business models and process models. The authors define business models as means to describe actors and their value exchange, whereas process models define for each actor how to realize value exchanges. Subsequently, they analyze the relationship between the two models and propose a formalization for each model. The findings are that business models relate to UN/CEFACT UMM and that process models relate to ebXML BPSS.

Likewise, Andersson et al. [1] distinguish between business models and process models in the e-commerce domain. They associate business models with business analysis, and ascribe process models with low-level activities and their ordering. Andersson et al. find evidence for a relationship between the two models and propose a systematic method to generate process models from business models. The presented formalism for business models refers to the  $e^3$  Value approach (an approach to evaluate e-commerce ideas) [6], whereas process models are described as patterns which origin in UN/CEFACT UMM. The outlined routine has five steps: (1) Start with a  $e^3$  Value model, (2) check custody, (3) check evidence, (4) identify a set of processes, and (5) for each process, select a pattern from the UMM.

Dorn et al.'s work [5] targets business-related and technical-related specifications in the business-to-business e-commerce domain. They acknowledge a relationship between these two types of specifications. Furthermore, a survey shows available

specifications and their possible overlaps. This survey is based on a refinement of the open-EDI reference model, which distinguishes two views: (1) Business Operation View (BOV) and (2) Functional Service View (FSV). They refine the BOV into business models and process models and the FSV into deployment artifacts and software environments. This refined model groups existing methodologies and technologies. Lastly, they suggest a methodology to design e-commerce applications, starting with the design of business models, developing business processes, deriving system architectures, and finally implementing e-commerce applications.

Likewise to the ISE framework, all these approaches acknowledge the existence of a business model layer and a conceptual layer, which are both technology-agnostic. The business model layer comprises strategic aspects and addresses organizations as well as their suppliers, customers, and competitors, whereas the conceptual layer targets individual organizations and the internal configuration of the value creation process. Unlike the ISE framework, all of these approaches focus mainly on business process automation and are limited to behavioral aspects. The differences between the proposed idea in this paper is that it targets on service engineering and incorporates processes as well as descriptions for rules, data, human resources, and services' value.

### III. THE ISE FRAMEWORK

Based on a state-of-the-art study of existing frameworks, Kett et al. [8] argued that existing frameworks for service engineering either address the business perspective or the technical perspective. To overcome the gap between these approaches they introduced the Inter-enterprise Service Engineering (ISE) framework (cf. figure 1), a framework for service ecosystems, which embraces the Zachman framework [22] and a service engineering methodology for service products [3].

The horizontal axis shows four perspectives of the engineering process and is named *abstraction layers*. Each perspective relates to a specific role with appropriate skills and offers different sets of tools and methods. It also implies the chronology of the framework. Additionally, the perspectives are linked to phases of the service engineering process. The vertical axis (dimensions) shows five different descriptions of a service. Each description is valid for each perspective. Any intersection in the matrix is placeholder for a model, a notation, and a modeling technique, which is appropriate for the respective perspective and the modeling aspect.

#### A. Dimensions

The *service description* dimension embodies services' value proposition toward potential customers. This includes functional, financial, legal, marketing, and quality of service properties as well as other meta data for service proposition, discovery, selection, contracting, and monitoring. The *workflow* dimension addresses services' behavioral aspect, which include core capabilities and sequence flows. The *people* dimension offers means to model and to refine human resources, and to assign tasks. Intangible assets, terms, and

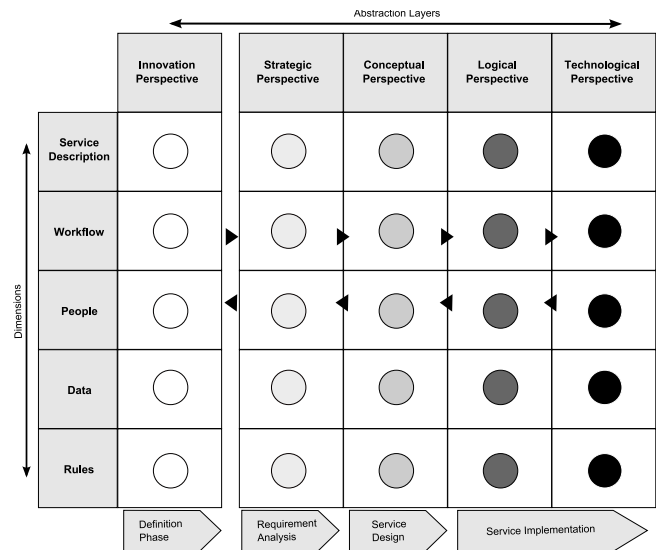


Fig. 1. ISE Framework [8]

concepts as well as their relationships are defined in the *data* dimension. The *rules* dimension addresses structural and organizational rules. These are defined to elicit and formalize domain knowledge to guide services' behavior.

#### B. Perspectives

The *innovation* perspective is out of scope of the ISE framework. It marks the interface to service innovation [9] and defines a first proposal for a new service. Business strategists pick up new service ideas and focus on requirement analysis in the *strategic* perspective. Kett et al. [8] depicted a basic underlying model for this perspective: the Business Model Ontology (BMO) [14]. Eventually, a decision is made whether to implement a new service or not. The *conceptual* perspective focuses on operationalizing and implementation of strategic artifacts from the owner's perspective. Proposed modeling notations were Business Process Modeling Notation (BPMN) [11], Unified Modeling Language (UML), and Event-driven Process Chains (EPC) [17] to transform domain experts' perceptual requirements into appropriate models. The final artifact is a service design which is neither technical nor platform-specific. Conceptual artifacts are transformed into abstract technical models during the *logical* perspective by IT analysts. This perspective offers a bridge between service design and technical service implementation. Finally, the IT developer transforms the logical artifacts into platform-dependent software artifacts, e.g., WSDL, SOAP, etc., during the *technical* perspective.

### IV. A CLEAR SEMANTIC FOR THE STRATEGIC AND CONCEPTUAL PERSPECTIVE

This section advances the semantic for the strategic and conceptual perspective in the ISE framework in two aspects in order to ease the framework's application. First, new names are proposed in order to establish term coherence and to reduce ambiguity. Second, to advance the semantics for the

descriptions and for two perspectives, additional concepts are proposed. Furthermore, for each cell do exist meta-models which are not shown here due to space restrictions.

The term *service description* in the ISE framework is revised into *value description* in order to avoid ambiguity: a service is described by the union of all descriptions, whereas the value description is restricted to services' propositions toward potential customers (cf. [6], [18]). The term *people description* is revised into *actor description* to stress the point that this description includes next to human beings also companies as well as governmental institutions. The term *workflow description* changes into *process description* which relates to the Zachman framework. Table I shows the resulting framework together with its formalization concepts.

#### A. Strategic Perspective

The strategic perspective utilizes different concepts from BMO [14], which was already motivated by Kett et al. [8]. BMO is an ontology with nine abstract concepts to accurately describe companies's business models. However, novel in the strategic perspective is the application of OMG's Business Motivation Model (BMM) [10] for the rule description. BMM offers a coherent scheme to manage and communicate business plans and a government structure which relates to business rules. Additionally, the  $e^3$  Value approach from Gordijn [6] is also considered for the value description.

*Value Description:* Kett et al. [8] proposed the following concepts from BMO [14]: (1) value proposition, (2) distribution channel, (3) relationship, and (4) revenue model. However, considering that the value description is targeted at service consumers, the target customer concept from the BMO is added as well, which is also supported by Gordijn [6].

*Data Description:* A data description depicts a shared terminology within companies for resources which are input and/or result of process activities. It is important to note that the terms data and resources are synonymical in the context of this work. Kett et al. [8] proposed BMO's immaterial resource concept [14]. In contrast to Kett et al., this work considers also material resources in order to represent a complete terminology. Relationships between resources, however, are omitted and firstly considered in the conceptual perspective.

*Actor Description:* The strategic perspective's meta model uses the BMO concepts partnership, actors as well as capabilities. Both, the actor node as well as the capability node embody the attributes name and description. Actor and capability nodes may be linked in order to declare which actors provide what capability. Yet, the actor node is a general concept and is refined by the partner node and the role node. The partner node depicts *outside* organizations whereas the role node represents human resources from *inside* organizations.

*Process Description:* Kett et al. [8] proposed to use the value configuration and the capability concept from BMO. A subtle refinement is made here: capabilities are modeled within the actor description since the capability concept has a close relationship with actors. Additionally, the BMO's activity concept is added to the process description.

TABLE I  
REVISED FRAMEWORK

	Strategic Perspective	Conceptual Perspective
Value Description	BMO [14] $e^3$ Value [6]	Service Properties [18]
Process Description	BMO [14]	BPDM [12]
Actor Description	BMO [14]	Org. Charts [21]
Data Description	BMO [14]	ERM [4]
Rule Description	BMM [10]	SBVR [13]

*Rule Description:* As aforementioned in this section, the meta model for the strategic perspective is motivated by parts of BMM [10]. According to OMG, *ends* represent anything organizations seek to achieve, whereas *means* refer (among other things) to instruments to *realize* ends [10]. Hence, according to the definition by Kett et al. [8], the ends concept fits into the strategic perspective and the means concept into the conceptual perspective.

#### B. Conceptual Perspective

The conceptual perspective takes advantage of existing specifications and available approaches. Contrary to the strategic perspective, meta models in this perspective are richer as well as more expressive for this perspective focuses on information and its interrelation.

*Value Description:* Service offers in the conceptual perspective reflect a firm establishment with concrete values. The meta model for the conceptual perspective builds on two approaches. Scheithauer & Winkler [19] investigated properties to describe services to allow service offering, discovering, selection, and consumption. Scheithauer et al. [18] propose a meta model for these properties and their relationships (cf. table II).

*Data Description:* In the conceptual perspective a business terminology is refined into a fact model. A fact model expands a terminology with resources as well as attributes. As aforementioned, relationships interrelate resources, which in turn represent business-relevant knowledge. Zur Muehlen et al. [23] refer to this knowledge as *structural rules* whereas the Business Rule Group defines interrelated resources as *facts*. Additionally, it resembles the Entity-Relationship diagram [4].

*Actor Description:* The conceptual perspective's meta model resembles the strategic's meta model with the difference that the employee node is added to the model and is similar to the organizational chart model depicted in [21]. The node has merely a name attribute and may link to roles in case an employee matches a role's profile as well as directly to capabilities.

*Process Description:* Evidence for the conceptual process model can be found in OMG's Business Process Definition Metamodel (BPDM) [12], which offers a meta model for business processes in order to compare and align different process notations, such as BPMN and EPC [17].

*Rule Description:* Contrary to the strategic perspective, the conceptual perspective utilizes the *means* concept, that is how to accomplish defined goals and objectives. The BMM specification [10] offers business rules to support objective’s achievement. The introduced rule concept for the conceptual perspective is informed by the work of the Semantics of Business Vocabulary and Rules (SBVR) specification [13]. It is important to note that the business rule concept relies on the data description with its resources which relate to the SBVR’s *business vocabulary*.

## V. INTEGRATED PROCEDURE MODEL

This section introduces an initial procedure model for the ISE framework. It aims at bridging the strategic and conceptual perspectives by means of eleven abstract steps that contain fine-granulated activities (cf. figure 2). This procedure model is influenced by work of zur Muehlen et al. [23]. They offered an abstract procedure model for integrated process and rule modeling. This work is extended for the integrated procedure model for all descriptions spanning the strategic and the conceptual perspective. The steps one to five address the strategic perspective, whereas the steps six to eleven address the conceptual perspective. The following subsections elaborate on each abstract step. Likewise zur Muehlen et al. [23] each step is explained by the triple: activities, challenges, and results.

### A. Strategic Perspective

The first five steps support business strategists when transforming a service idea (service innovation perspective) into a tangible foundation for business strategists, business analysts as well as business owners to decide whether to implement a service or not (predetermined breaking point).

1) *Define Value Offer:* The first step includes to define a value offer, which is an abstract service description. All following steps in this perspective take this outcome as a requirement document. *Activities.* The activities for this step include: (1) Establish exactly one value proposition, (2) determine one or more target customers, (3) determine exactly one relationship for each target customer, (4) determine one or more distribution channels, and finally (5) setup one or more revenue models. *Challenges.* In order to identify the concepts for this model, a deep understanding of the business domain as well as marketing is necessary. *Result.* The outcome of this step is an instance of the model described in section IV-A: a value offer which describes the service from a strategic perspective. The artifact serves as a requirement for the following steps in the strategic perspective.

2) *Determine Key Business Activities:* Once the value offer artifact is provided, business strategists determine a value configuration type as well as business activities. The value configuration type implies the nature of the value configuration; whether the value creation process compares to a value chain, a value shop, or a value network. *Activities.* The activities include to determine all necessary business activities to fulfill the *value offer* (cf. [14]). *Challenges.* Business strategists must

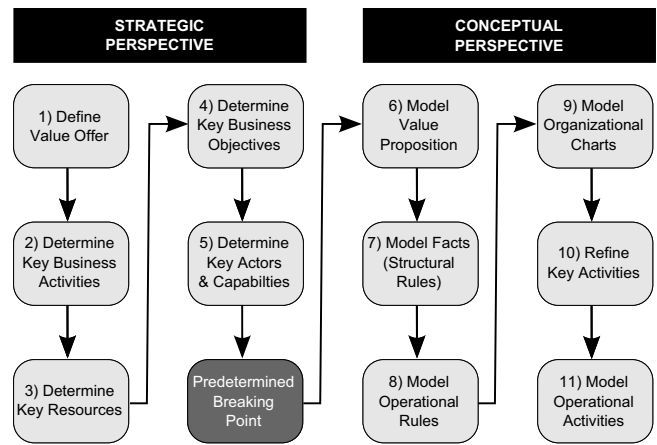


Fig. 2. Procedure Model for Business Service Modeling

understand the business domain’s value creation process as well as the own company to determine the value configuration type and necessary business activities. *Result.* The outcome of this step is a value configuration, which determines the process description from a strategic perspective.

3) *Determine Key Resources:* After the determination of crucial business activities, business strategists need to identify resources which are required and/or produced by these activities. These resources also serve as a general terminology for the service to be developed. In the strategic perspective the terms *business object*, *term*, and *resource* are synonymic. This step refers to the step *Determine Key Objects, Terms, and Definitions* from zur Muehlen et al. approach [23]. *Activities.* The single activity for this step is to identify resources which are needed for determined business activities, to provide each resource with a name and a textual description, and to specify whether it is a tangible or intangible resource. *Challenges.* It is important to know the scope and the implications of each business activity in order to determine appropriate resources. *Result.* The outcome of this step is a business terminology, which determines the data description from a strategic perspective.

4) *Determine Key Business Objectives:* Business rules codify behavioral business knowledge; things companies want to accomplish. BMM offers three elements: Whereas *visions* describe companies’ desired state in the future, *goals* refer to a qualified and long-term statement to achieve a vision. Finally, *objectives* refer to a quantified (measurable) and short-term statement to achieve a goal. *Activities.* The activities to determine business objectives are a refinement process. It starts with establishing visions. For each vision one or more goals must be determined. Finally, attainable, time-targeted, and measurable objectives need to be derived for each goal [10]. *Challenges.* Developing business objectives is a non-trivial part with far-ranging implications. It affects services’ internal behavior and routes the value creation process. *Result.* The outcome of this step is a business objective terminology, which determines the rule description from a strategic perspective.

5) *Determine Key Actors & and their Capabilities*: It is necessary to model actors after the process descriptions, for each business activity needs to be performed by either a partner or companies' personnel. *Activities*. For each business activity from the process description one or more capabilities need to be identified in order to accomplish the activity. Following this, business strategists identify personnel with appropriate capabilities. Likewise, other organizations must be identified for crucial capabilities which cannot or rather should not be achieved internally. *Challenges*. This step's peculiarity is to derive appropriate capabilities for all business activities. Business strategists need to be aware of their own personnel and its capabilities. Additionally, they need to identify appropriate partners for the service's realization and decide what kind of partnership they want to establish. *Result*. The outcome of this step is a set of business actors, which determines the actor description from a strategic perspective.

### B. Conceptual Perspective

The last six steps of the integrated procedure model deal with the conceptual perspective (cf. Figure 2). These steps support business analysts to conceptualize a service, hence determine *how* to implement a service strategy. The outcome of this perspective serves as a means for communication and a support for decisions. It is neither technical nor platform-dependent. Furthermore, it is a starting point for the transformation into technical specifications [20].

6) *Model Value Proposition*: In this step the strategic perspective's value model is refined. One goal is to operationalize aspects from the strategic perspective into a value proposition which is available to potential customers. Hence, all strategic artifacts with internal knowledge must be revealed, including the value level, the customer equity, and the revenue model. *Activities*. The general order of value modeling is: (1) Functionality, (2) Quality, (3) Marketing, (4) Legal, and finally (5) Finance. Table II shows categories and their properties (cf. [18]). Corresponding to each property are entities from the strategic perspective's value model. This light-weight mapping between the strategic and the conceptual perspective eases the value description modeling. *Challenges*. The challenges involved in value modeling are manifold, since business analysts need knowledge in the domains of marketing, quality of services, pricing mechanisms, and legal aspects. *Result*. The outcome is an instance of a value model (cf. [18]).

7) *Model Facts*: Business analysts augment identified resources in the strategic perspective with attributes and relations. *Activities*. For each resource business analysts make out attributes to describe resources sufficiently. Following this, they model relationships between resources. These relationships are also named *facts* or *structural rules*, that is, knowledge between two or more resources. Finally, meaningful attributes are added to each relationship. *Challenges*. The challenge is to ensure completeness for the fact model. *Result*. The outcome is an instance of a fact model (cf. [4]).

8) *Model Operational Rules*: Modeling operational rules implies to constrain facts in such a way that it guides services'

TABLE II  
MAPPING BETWEEN STRATEGIC & CONCEPTUAL VALUE DESCRIPTION

Category	Property [18]	Influenced by Strategic Entities [14]
1. Functionality	Capability	Value Offer Customer Equity
	Classification	Value Offer Target Customer
2. Quality	Performance	Value Offer Price Level
	Dependability	Value Offer Price Level
3. Marketing	Certification	Value Offer Target Customer Revenue Model Distribution Channel
	Expert Test Rating	Value Offer Target Customer Revenue Model Distribution Channel
	Benefit	Value Offer Target Customer
4. Legal	Right	Value Offer Target Customer Customer Equity
	Obligation	Value Offer Target Customer Revenue Model Distribution Channel
	Penalty	Target Customer Revenue Model
5. Finance	Price	Revenue Model Customer Equity Target Customer Value Level Price Level
	Discount	Value Level Price Level
	Payment	Target Customer Life Cycle Step

internal behavior according to business objectives (cf. [23]). *Activities*. Business objectives as well as the fact model serve as a basis for this step. Business analysts augment facts (relations between resources) with constraints in order to codify (formalize) business knowledge which supports coherent decision making in business processes. As aforementioned, SBVR [13] is suitable for this step. *Challenges*. The gist lies in transforming business objectives into constraints, and hence to operationalize them. *Result*. The outcome of this step is a fact model augmented with operational rules.

9) *Model Organizational Charts*: Business analysts refine the personnel in this step. *Activities*. The personnel element from the strategic's actor description is refined into roles with capabilities. Available employees (person element) are categorized into roles according to their individual capabilities. *Challenges*. Business analysts need to match required capabilities with employees' individual capabilities. *Result*. The outcome of this step is an organizational chart describing roles with considered capabilities and assigned employees.

10) *Refine Key Activities: Activities.* Business analysts refine strategic perspective's business activities into supporting business processes with a lower granularity (cf. [23]). *Challenges.* The challenge is to identify completely all necessary processes as well as to control the granularity level [23]. *Result.* The outcome of this step is a set of fine granular business processes for each business activity.

11) *Model Operational Activities:* The final step in the procedure model is to combine actors, rules, data, and processes by specifying operational activities. Operational activities are assumed to be atomic, and thus, cannot be decomposed into fine granulated activities and can either be assigned to a specific role inside a company or to a partner. *Activities.* For each business process, business analysts start by modelling each actor who is involved in the business process. Following this, they use operational activities, events, gateways, roles, conditional flow, sequence flow, and message flow elements to define services' internal behavior. Additionally, data flow combines activities and resources and depicts activities's inputs and outputs. *Challenges.* Business analysts need sophisticated knowledge about operation activities [23]. *Result.* The final outcome of this step is a complete set of business processes combining activities, rules, and data.

## VI. CONCLUSION & FUTURE WORK

A recent study [15] shows that existing software engineering methodologies do not apply to service-oriented design due to highly dynamic environment, high uncertainty, distributed control of processes, many different stakeholders, and finally that decisions cannot be foreseen during design time, which holds also true for service ecosystems and its peculiarities. Therefore, the Inter-related Service Engineering (ISE) framework [8] was introduced, which offers a methodology for service-oriented engineering. Three areas for improvements were identified. The first improvement addresses a firm terminology in that it reduces term ambiguity. The second improvement proposes additional concepts and meta models to advance the framework's semantic. The last improvement presents an initial integrated procedure model with eleven steps, which will guide the modeling process.

Business information science benefits from the incorporation of actual studies in the areas of business service modeling and service engineering in that it interconnects popular modeling notations. Furthermore, the procedure model reduces the framework's complexity and enables industries to apply the framework.

This work's major limitation is a missing verification of the procedure model. This issue will be addressed in the next step of the Theseus/TEXO research project [7]. Additionally, future work also includes to advance the procedure model for the logical and technical perspective. Ideas found [20] present potential for improvements in this direction.

## ACKNOWLEDGMENTS

This project was funded by means of the German Federal Ministry of Economy and Technology under the promotional

reference "01MQ07012". The responsibility for the content of this publication lies with the authors.

## REFERENCES

- [1] ANDERSSON, B., BERGHOLTZ, M., GRÉGOIRE, B., JOHANNESON, P., SCHMITT, M., AND ZDRAVKOVIC, J. From Business to Process Models - a Chaining Methodology. In *BUSITAL* (2006), Y. Pigneur and C. Woo, Eds., vol. 237 of *CEUR Workshop Proceedings*, CEUR-WS.org.
- [2] BERGHOLTZ, M., JAYAWEEERA, P., JOHANNESON, P., AND WOHEDE, P. Process Models and Business Models - A Unified Framework. In *ER* (2002), S. Spaccapietra, S. T. March, and Y. Kambayashi, Eds., vol. 2503 of *Lecture Notes in Computer Science*, Springer, pp. 364–377.
- [3] BULLINGER, H.-J., FAHRNICH, K.-P., AND MEIREN, T. Service Engineering – Methodical Development of new Service Products. *Int. Journal of Production Economics* 85, 3 (September 2003), 275–287.
- [4] CHEN, P. P. The Entity-Relationship Model - A basis for the Enterprise View of Data. In *AFIPS National Computer Conf.* (1977), pp. 77–84.
- [5] DORN, J., GRUN, C., WERTHNER, H., AND ZAPLETAL, M. A Survey of B2B Methodologies and Technologies: From Business Models towards Deployment Artifacts. In *HICSS* (2007), IEEE Computer Society, p. 143.
- [6] GORDIJN, J.  $E^3$ -value in a Nutshell. Tech. rep., HEC University Lausanne, Lausanne, Oct. 07 2002.
- [7] JANIESCH, C., RUGGABER, R., AND SURE, Y. Eine Infrastruktur für das Internet der Dienste. *HMD - Praxis der Wirtschaftsinformatik* (45:261), 2008, pp. 71–79, June 2008.
- [8] KETT, H., VOIGT, K., SCHEITHAUER, G., AND CARDOSO, J. Service Engineering for Business Service Ecosystems. In *Proceedings of the XVIII. Int. RESER Conf.* (Stuttgart, Germany, September, 25 - 26 2008).
- [9] MAGLIO, P. P., SRINIVASAN, S., KREULEN, J. T., AND SPOHRER, J. Service Systems, Service Scientists, SSME, and Innovation. *Communications of the ACM* 49, 7 (July 2006), 81–85.
- [10] OBJECT MANAGEMENT GROUP (OMG). Business Motivation Model (BMM), Adapted Specification. August 2006.
- [11] OBJECT MANAGEMENT GROUP (OMG). Specification: Business Process Modeling Notation (BPMN), Version 1. Feb 2006.
- [12] OBJECT MANAGEMENT GROUP (OMG). Business Process Definition Metamodel (BPDm), version 1.0. November 2008.
- [13] OBJECT MANAGEMENT GROUP (OMG). Specification: Semantics of Business Vocabulary and Rules (SBVR), Version 1.0. January 2008.
- [14] OSTERWALDER, A. *The Business Model Ontology: A Proposition in a Design Science Approach*. PhD thesis, Université de Lausanne Ecole des Hautes Etudes Commerciales, 2004.
- [15] PAPAIOGLOU, M. P., TRAVERSO, P., DUSTDAR, S., AND LEYMAN, F. Service-Oriented Computing: a Research Roadmap. *Int. J. Cooperative Inf. Syst.* 17, 2 (2008), 223–255.
- [16] PENEDER, M., KANIOVSKI, S., AND DACHS, B. What Follows Tertiarisation? Structural Change and the Role of Knowledge-based Services. *The Service Industries Journal* 23 Issue 2, 146 (March 2003), 47–66.
- [17] SCHEER, A.-W., AND NUETTGENS, M. Architecture and Reference Models for Business Process Management. *Lecture Notes in Computer Science* 1806 / 2000 (2000), 376–389.
- [18] SCHEITHAUER, G., AUGUSTIN, S., AND WIRTZ, G. Describing Services for Service Ecosystems. In *ICSOC Workshops* (Sidney, Australia, December, 1 2008), G. Feuerlicht and W. Lamersdorf, Eds., vol. 5472 of *Lecture Notes in Computer Science*, Springer, pp. 242–255.
- [19] SCHEITHAUER, G., AND WINKLER, M. A Service Description Framework for Service Ecosystems. *Bamberger Beiträge zur Wirtschaftsinformatik* 78, Bamberg University, October 2008. ISSN 0937-3349.
- [20] SCHEITHAUER, G., WIRTZ, G., AND TOKLU, C. Bridging the Semantic Gap between Process Documentation and Process Execution. In *The 2008 Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'08)* (Redwood City, California, USA, July, 1 - 3 2008).
- [21] WESKE, M. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, Berlin, 2007.
- [22] ZACHMAN, J. A. A Framework for Information Systems Architecture. *IBM Systems Journal* 26, 3 (1987), 276–292.
- [23] ZUR MUEHLEN, M., INDULSKA, M., AND KITTEL, K. Towards Integrated Modeling of Business Processes and Business Rules. In *19th Australian Conf. on Information Systems ACIS 2008* (Christchurch, New Zealand, December, 3–5 2008).



# A Systematic SOA-based Architecture Process

José Jorge Lima Dias Júnior<sup>1</sup>, Eduardo Santana de Almeida<sup>3</sup>, Silvio Romero de Lemos Meira<sup>2,3</sup>  
*DATAPREV – Technology and Information Company of Social Security*<sup>1</sup>  
*Federal University of Pernambuco*<sup>2</sup>  
*C.E.S.A.R. - Recife Center for Advanced Studies and Systems*<sup>3</sup>  
*jose.jorge.jr@previdencia.gov.br, esa@rise.com.br, silvio@cesar.org.br*

## Abstract

During recent years, the notion of software architecture has emerged as the appropriate level for dealing with software quality. This sub-discipline of the software engineering has a several foundations that characterize a set of aspects in the architecture processes, such as views-oriented description, quality attributes orientation and architecture evaluation. Service-Oriented Architecture (SOA) emerged as a type of software architecture to build systems through the composition of services. On the one hand, the traditional architecture processes do not comprise some SOA features. On the other hand, the available SOA approaches do not fulfill all the software architecture foundations. In this sense, this paper proposes a systematic SOA-based architecture process that complains the main software architecture foundations and SOA features in order to guide the architects in the construction of a software architecture description.

**Keywords:** Architecture Process, SOA, Software Architecture.

## 1. Introduction

Software Architecture has attracted a great attention from researchers and practitioners since the last decade. The increasing size, complexity, and demand for quality software systems are some of the drivers that have increased interest in this sub-discipline of software engineering [7].

As any activity of software engineering, it is useful to follow a defined process in order to guide the architect through the definition of the application architecture [14]. A complete architecture process can include three main activities [14], [15]: *define the architecture requirements*; *design the architecture*; and *evaluate the architecture*. Moreover, the architecture documentation is part of the architecture design. However, it is common to find these activities separated in different methods in the literature. For example, some methods [2], [20], [25] focus on the architecture design activities and other methods [18], [19] focus on the architecture evaluation. The first one is concerned with the creation of software architecture, and the second one aims at analyzing a software architecture in order to identify potential risks and verify if the quality requirements have been addressed in the architecture.

An architecture design method aids the architects to design a software architecture description, considering styles that can be used and different views for several goals, addressing different quality attributes. Nevertheless, architecture design methods that were developed in different domains naturally exhibit domain characteristics and emphasize different goals [15]. Therefore, none of these methods [2], [20], [25] alone is comprehensive enough to cover the design of software architectures for systems with different sizes on various domains [21].

Currently, a special type of software architecture is being widely explored by academy and industry: Service-Oriented Architecture (SOA). A basis of the SOA is the concept of service as a functional representation of a real world business activity meaningful to the end-user and encapsulated in a software solution [26]. In the enterprise context, SOA allows the organizations, which have a highly fragmented application infrastructure under management of different business areas, integrate these applications in the service level [10].

On the one hand, the traditional architecture processes do not comprise these SOA features. On the other hand, the available SOA approaches [1], [11], [12], [17], [23] do not fulfill all the software architecture foundations, such as quality attributes orientation, views-oriented description and architecture evaluation.

In this sense, this paper proposes a new process to create an SOA-based architecture description. For this purpose, the main activities in the software architecture processes and the main features found in the SOA approach were elicited. Furthermore, an initial evaluation was performed in order to assess the proposed process, in which the difficulties of its use, the guidance that it provides to the architects and the completeness of the architecture description created were analyzed.

The remainder of this paper is organized as follows: Section 2 discusses the foundations of the proposed process. Section 3 presents a new SOA-based architecture process. Section 4 presents the evaluation performed to assess the proposed process. Section 5 discusses related work. Finally, Section 6 presents some concluding remarks.

## 2. Process Foundations

A software development process can be understood as the set of activities needed to transform the user's

requirements into a software system. The way it is done can change from process to process. In this sense, all types of processes are based on some foundations. In this section, the foundations used to elaborate the process are presented.

**Quality Attribute Oriented.** Quality attribute requirements must be considered early in the life cycle, and the software architecture must be designed so that their quality attributes are met [3]. Hence, the process supplies a specific activity to elicit these requirements in order to address them through different views.

**Views-oriented Description.** Software architecture is a complex entity that cannot be described in a simple one-dimensional fashion [4]. A view is a “*representation of a whole system from the perspective of a related set of concerns*”, and a viewpoint is a “*specification of the conventions for constructing and using a view*” [16]. Thus, the process proposes viewpoints to represent different concerns of the SOA.

**Architecture Evaluation.** Since software architecture plays a significant role in archiving system wide quality attributes, it is very important to evaluate a system’s architecture with regard to desired quality requirements [8]. In this sense, the process proposes a specific activity for evaluating the SOA in order to verify if the architectural decisions are addressing the quality attributes. However, an existent method is suggested in this activity.

**Business process oriented.** Services provide a better way to expose discrete business functions and therefore a good way to develop applications that support business processes [6]. In this context, the process considers and analyzes the enterprise business processes in order to identify the services of the SOA.

**Design by contract.** An important aspect of the SOA is that it separates the service’s implementation from its interface. To successfully use a service, both the consumer and provider need to understand the contract — what the implementation agrees to do for the consumer [22]. Thus, the process enables the definition of the service contract.

**Service-orientation principles.** The service-orientation principles are considered in the service interface definition and other design activities in order to maintain the SOA foundations.

**Multiple development teams.** One of the main benefits of using a SOA is that it allows a high degree of modularity. This feature permits that the services can be implemented for different teams [17]. Hence, a SOA-based process must be organized in order to enable the division of work in multiple teams to implement the services.

### 3. The Process

Along this section, how the foundations were attended by the proposed process will be explained.

The roles involved in the process were divided in two groups:

- **Enterprise business team:** Roles of people who have an overall view of the enterprise. This group is divided in

two roles: *Enterprise manager* and *SOA architect*. The first one is the person who knows the enterprise business as a whole and manages the integration of business area teams. The second one is a specialist that knows about SOA concepts and technologies.

- **Business area team:** Internal or external people of the enterprise that are interested on providing or consuming some service. This group can be divided in two subgroups: *Service development team* and *Business specialist team*. The first one has the roles related to services development, i.e., the team that will design and implement individual services. The second one has the roles of business specialists of some business area.

Development teams in a SOA-based enterprise project can be decoupled due to high degree of modularity by using a SOA. Thus, each team is responsible for implementing a specific list of services. These teams must focus on the agreement of service contracts [18].

In order to fulfill the foundation of having multiple development teams, the proposed process is composed by two phases: **SOA Definition** and **Service Design**. The first one has activities related to the architecture definition of the SOA-based enterprise system. The second one aims at designing the services identified in the SOA Definition phase. In this case, each service is designed by the team responsible for providing the service.

Figure 1 shows the two phases of the process, in which the Enterprise business team participates in the activities of the SOA Definition phase along with all Business specialist teams (A, B and C), and in the next phase of Service Design, in which each service development team is responsible for designing its own services.

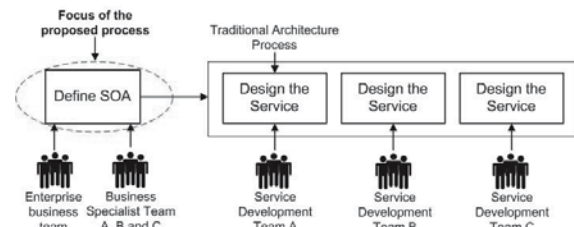


Figure 1. Phases of the proposed process.

This paper focuses on the SOA Definition phase. For this reason, this paper aims at detailing only this phase. In the Service Design phase, a traditional architecture process to design the individual services can be used.

The SOA Definition phase is composed by three activities. Briefly, the three activities are the following:

- **SOA Analysis:** The services are identified as well as the consumers and providers. Besides, the quality attributes of the SOA and of the specific services are also identified.
- **SOA Design:** The service contracts are specified and it is decided if these services will be reused or developed. Moreover, the views are created in order to address the quality attributes of the SOA and of the services, and the SOA documentation is produced.

• **SOA Evaluation:** The architecture is evaluated in order to verify if it is capable of fulfilling required quality attributes and to identify any potential risks.

The proposed process defines sub-activities for the first two activities and it uses an existent evaluation method for the third one. These activities will be seen in details in the next subsections.

### 3.1 SOA Analysis Activity

This activity has the objective to identify the services and the respective consumers and providers as well as the quality attributes for each service and for the SOA as a whole. The next subsections discuss the sub-activities of the SOA Analysis activity.

#### 3.1.1 Identify Service Interfaces

This is one of the most important concerns in service-oriented design. This sub-activity aims at uncovering the services that will compose the SOA. For this purpose, this work proposes the *Service Identification Workshop* (SIW).

The SIW provides an opportunity to join the areas to provide input about their needs and expectations with respect to services that are of particular concern to them. Hence, business specialist team and enterprise business team will achieve meetings in order to identify the service interfaces. Figure 2 shows the steps of SIW.

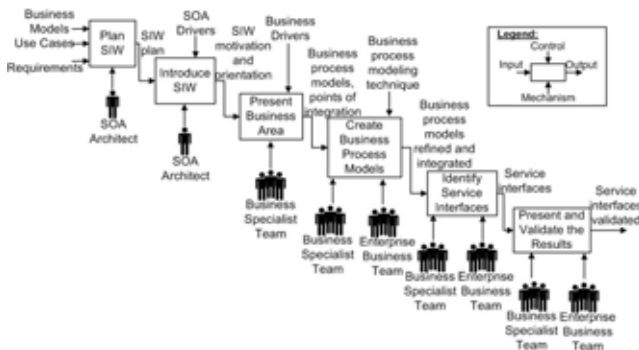


Figure 2. Steps of SIW.

Firstly, the SIW must be planned. For this purpose, the SOA architect analyzes the business areas, studying the macro characteristics of each one. In this sense, he can analyze documents available about the business area, existent systems, and new systems such as business process models, requirements and use cases. Next, the workshop can be started with the SIW introduction. In this step, the SOA architect explains the motivation for the SIW and describes each step of the technique for the participants. Moreover, the stimulus to use SOA for the enterprise must be presented, showing the benefits of this architecture solution. After that, each business specialist team must present its business area, showing the main business processes, responsibilities, functionalities of existent and new systems, and so on. Besides, each team must take clear about its business drivers for using services of another business area. The next step is *Create Business Process Model*. The idea of this step is to create new business

processes or remodel the existent ones of each business area in order to represent them in the most granular representation of processing steps. This idea is the same found in the Erl's approach [11] in which it aims at taking the business process and breaking it down into a series of granular process steps in order to identify the services based on them. In addition, these business processes must represent the points of integration among the business areas identified in the prior step. Since the business process models have been created or refined, the services can be identified. This step is also conducted by the SOA architect that along with the business specialist teams will decide what business activities identified in previous step will be considered software services. After consolidation of the services identified, the SOA architect must present them to all the SIW participants in order to validate the results.

#### 3.1.2 Categorize Service Interfaces

Services have different uses and purposes [26]. Hence, services can be classified through their operational characteristics with different granularity levels [24]. Moreover, to categorize the services according to their purpose, increase the potentiality of reuse [11]. Currently, diverse service taxonomies are found in the literature [11], [12], [17], [24]. The SOA architect must choose some taxonomy and categorize the services identified in the previous sub-activity. This choice of the taxonomy depends on how the architect intends to organize the services. In this sense, the categorization of the services will influence the organization of the SOA layers in the SOA Design activity.

In this work, the taxonomy presented by Papazoglou [24] was used. It uses three categories as following:

- **Business Services:** They automate a generic business task with significance to the business process.
- **Technical Services:** They are coarse-grained services that provide the technical infrastructure enabling the development, delivery, maintenance and provisioning of business processes.
- **Utility Services:** They are fine-grained services that provide value to business services across the organization, for example, services implementing calculations, algorithms, directory management services, etc.

The output of this sub-activity is the service interfaces categorized in accordance with this taxonomy.

#### 3.1.3 Apply Service-Oriented Principles

It is important that principles be applied to the services interfaces so that they have characteristics inherent to the service-oriented approach.

There is no official set of service-orientation principles. Diverse principles related to service-orientation approach are found in the literature. Papazoglou and Heuvel [23] and, Feuerlicht and Lozina [13] list three principles: *service coupling*, *service cohesion* and *service granularity*.

The service coupling can be seen as the degree of interdependence among two business processes. Hence, the objective is to minimize coupling, i.e., to make business

processes as independent as possible by not having any knowledge of or relying on any other business processes.

Cohesion refers to the level of interrelationships among the elements of a software module. High level of service cohesion increases application stability as cohesion limits the impact of changes to a small number of services.

Service granularity refers to the scope of functionality exposed by a service. Services may exhibit different levels of granularity. From the perspective of service-oriented design, it is preferable to create higher-level, coarse-grained interfaces that implement a complete business process.

The output of this sub-activity is the interfaces refined.

### 3.1.4 Identify Quality Attributes

After uncovering the service interfaces and their respective consumers and providers, it is necessary to identify the quality attributes for each service identified as well as for the SOA as a whole. Quality attributes could be missing from the requirements document, and even if addressed adequately, they are often vaguely understood and weakly articulated [3]. For this goal, an adaptation of *Quality Attribute Workshop* (QAW) [3] is used. QAWs were elaborated by SEI (Software Engineering Institute) and provide a method for identifying a system's architecture critical quality attributes. For this purpose, QAW generates, prioritizes and refines quality attribute scenarios before the software architecture is completed

The output of this sub-activity is the scenarios and a table with the quality attributes prioritized of each service and of the SOA as a whole. This prioritization is important since there are tradeoffs among the quality attributes. For example, if the system requires security then the performance is decreased.

## 3.2 SOA Design Activity

In this activity, the solution of the problem will be elaborated, specifying the service contracts, applying different viewpoints for the quality attributes identified and producing the architecture documentation. This activity was also subdivided in sub-activities.

### 3.2.1 Specify Service Contracts

Since the service interfaces with their respective potential consumers and providers are known, it is necessary to define the contract among them. A service contract contains the terms agreed by the service provider and service consumer for the supplying of the service.

In summary, the service contract must contain the following information: *service interface, service messages structure, pre- and post conditions, quality attributes, potential consumers, provider and SLA.*

This service contract can be specified with a simple document or a formal description such as WSDL specification.

### 3.2.2 Reuse Existent Services

The idea of this sub-activity is to search the services identified with their respective quality attributes in order to

verify if they already exist. In this case, the available services can be reused and the non-existent services will be developed. The providers for these services can be found in the internal or external area of the enterprise. Furthermore, it is necessary to decide how the services will be acquired (leasing, buying, and so on).

The output of this sub-activity is a table containing a set of services identified and the information if they will be reused or developed. In case of the service to be reused, additional information of the provider must be described, such as reputation, support, business model, target market, and so on [10].

### 3.2.3 Address Quality Attributes

Since the services and quality attributes were identified in the SOA Analysis activity, they must be addressed through different views. In this sense, for each identified service and for SOA as a whole, several views will be applied to represent these different concerns.

This paper proposes a set of viewpoints to represent the architecture decisions of the SOA. However, these viewpoints are not a closed list, and other viewpoints can be used according to demand of the SOA architect.

**Layer Viewpoint.** The tiers provide a conceptual structure at the enterprise level that organizes the services [18]. The objective of this viewpoint is to represent the layers of the architecture.

**Integration Viewpoint.** This viewpoint describes how the services will be integrated. The two significant options for a primary integration pattern are direct point-to-point and hub-and-spoke [5].

**Security Viewpoint.** Security can be achieved on different levels in the SOA approach, such as transport layer and message levels. Hence, the security view should represent the solution using one of these levels.

**Interaction Viewpoint.** The implementation alternatives impact important quality attributes of the system, such as interoperability and modifiability [5]. This view must represent how the consumer and the provider will be interacted.

**Physical Viewpoint.** Issues about interoperability, security and reliability can be decided, since it aims at capturing the distribution of the applications and services that compose the SOA as well as the transport and message protocol used in the communication among them.

**Registry Viewpoint.** This viewpoint aims at addressing how the services will be published, found and executed. The decision about the architecture of service discovery will be represented in this viewpoint.

**Publisher Viewpoint.** The services can be registered under diverse technologies such as UDDI, ebXML, P2P network, and so on. In this sense, this viewpoint must represent how the service will be published by the provider in order to turn it available to the consumers.

**Consumer Viewpoint.** This viewpoint must represent the following issues: how the consumer can find a service;

how the consumer acquires the service; and how the consumer can bind the service (dynamically or statically).

**Technical Process Viewpoint.** This viewpoint must represent the technical business processes that are part of the SOA, i.e., the business processes in the perspective of the services identified. These views are useful when it is necessary to map the technical business process into Business Process Modeling Language such as BPEL.

None of these viewpoints are mandatory. In this sense, the SOA architect must decide, according to the project, what viewpoints to use.

### 3.2.4 Produce SOA Documentation

After all activities have been realized, it is necessary to create the architecture documentation. This document is the main output of the architecture design activity, because it will document the artifacts produced.

All information that is necessary for the service development teams must be documented, since this documentation serves as a guide to them. Dias [9] presents the complete SOA documentation template.

### 3.3 SOA Evaluation Activity

Many architecture-centric analysis methods have been created in the few past years. Due to this variety of methods, an initial orientation is necessary. In this sense, the Architecture Trade-off Analysis Method (ATAM) [8] — developed by the SEI — is used as the basis for defining the activities for an architecture evaluation. For this purpose, SEI also produced a technical report about SOA evaluation [5] aiming at offering practical information to assist the evaluation of a system that uses the SOA approach.

This evaluation method was chosen because, beyond being a mature method, it is scenario-based and it can reuse the scenarios created in the QAW.

In spite of this activity being considered important, it is not mandatory to be performed. Moreover, other methods can be used in order to evaluate the architecture.

## 4. The Evaluation

The goal of the experiment was to analyze the proposed process for the purpose of evaluating it with respect to the difficulties of its use, the guidance that it provides to the architects and the completeness of the architecture description created from the point of view of the practitioners in the context of software architecture.

The experiment was run as an off-line project by professionals of a software development company. This company develops information systems for a government department that has five different areas that are responsible for some strategic business and have its own information system that helps to perform their business processes. Hence, this project is composed of five sub-projects running alongside that need to be integrated. Each sub-project has the objective to create an information system and each one has its own development team, with its project manager, software architect, business analyst, developers, and so on.

In this sense, the project of this experiment is a real problem of a SOA-based scenario with different development teams.

The experiment had the following results:

**Difficulties in the SOA Analysis activity.** The sub-activity *Apply Service-Oriented Principles* presented the most difficulty. Only one subject did not have problems to execute this sub-activity. It is important to highlight that the lack of experience of the subjects with SOA may have influenced this result. For example, the subjects considered difficult to perform this sub-activity because it does not specify the “step-by-step” to be executed. However, an expert SOA architect, having familiarity with these principles, would not have the same difficulty.

The experiment suggested that the SOA Analysis activity is difficult to be performed. However, it is necessary to highlight that this value for the null hypothesis was defined without any previous data, since it was the first time that this aspect was analyzed. Nevertheless, the next time that the experiment is performed this value can be refined based on this experience, resulting in a more calibrated metric.

**Difficulties in the SOA Design activity.** The main difficulty mentioned was to elaborate the views. Some factors can have influenced this result. First, the profile of the subjects, that had difficulty in this sub-activity, did not have experience in the software architecture position. Other factor can be related to the lack of experience in Web Services of some subjects.

The experiment suggested that the SOA Design activity is difficult to be performed. However, in the same way as in the SOA Analysis activity, this value for the null hypothesis was defined without any previous data.

**Guidance for creation of the architecture description.**

In spite of all the subjects suggesting that it is necessary to have guidelines for the sub-activities, they agreed that the sub-activities defined by the process guides the architect, since the process directs what must be produced in each stage of the process in the creation of the SOA description.

**Completeness of the architecture description.** Seven of eight quality attributes required were addressed by the architecture description. Only the quality attribute *Testability* was not addressed. In this sense, the creation of a *Test Viewpoint* was suggested in which it would contain information about how the services can be tested.

The experiment has evidenced that the process is useful in SOA context in order to guide the architect in the design of an architecture description. However, the experiment also shows that the process needs to be improved. Besides, other evaluations in other contexts must be performed to verify if the process can be applied on them.

## 5. Related Work

In this section, five known SOA approaches [1], [11], [12], [17], [23] were analyzed according to defined foundations elicited in the previous sections.

These approaches aim at supporting the full SOA lifecycle, including planning, analysis and design,

construction, testing, deployment, and governance activities, while others limit their scope to a subset of these phases, such as analysis and design [23]. For example, the Papazoglou's [23] covers all the development lifecycle and the Jones's approach [17] covers only initial planning. On the other hand, Erl's approach [11] covers only analysis and design activities and the SOAF [12] and SOMA [1], beyond analysis and design, covers the construction of the services. However, none of these approaches focus on the architecture activities, since they are not in accordance with some software architecture foundations.

In spite of all these approaches to consider the quality attribute aspects, none of them attend entirely and explicitly the view-oriented description and architecture evaluation requirements.

Regarding design by contract and service-orientation principles requirements, only the Jones's approach does not cover them. All the other approaches have some way to identify and to address the service contracts, and apply service-orientation principles. On the other hand, all of these approaches are focused in the business processes.

## 6. Concluding Remarks

A new SOA-based architecture process was proposed in this paper. It comprises the main important foundations of the software architecture and service-oriented areas such as service identification; quality attributes identification; service categorization; service contract specification; service reuse; SOA documentation; and SOA evaluation.

Moreover, an experimental study was performed in order to evaluate the proposed process. It was verified that the process needs to be improved, mainly in respect to guidelines to perform the sub-activities.

As a future work, a case study will be applied in practice to validate and refine the proposed process in a real project.

## Acknowledgments

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES<sup>1</sup>), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and Brazilian Agency (CNPq process number 475743/2007-5).

## References

- [1] Arsanjani, A. *Service-Oriented Modelling and Architecture (SOMA)*, IBM developer-Works, <http://www.ibm.com/developerworks/webservices/library/ws-soa-design1>, 2004.
- [2] Bachman, F., Bass, L. *Introduction to the Attribute Driven Design Method*. IEEE Software, 2001.
- [3] Barbacci, M. R., Ellison, R., Lattanze, A. J., Stafford, J. A., Weinstock, C. B., & Wood, W. G. *Quality Attribute Workshops*, 3rd. Edition. Pittsburgh, PA: SEI, Carnegie Mellon University, 2003.
- [4] Bass, L., Clements, P., Kazman, R. *Software Architecture in Practice*, Reading, Mass.: Addison-Wesley, 2003.
- [5] Bianco, P., Kotermanski, R. and Merson, P. *Evaluating a Service-Oriented Architecture*. SEI Technical Report, CMU/SEI-2007-TR-015, September, 2007.
- [6] Brown, A., Johnston, S., Kelly, K. *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications*. Rational Software Corporation, 2002.
- [7] Clements et al. *Documenting Software Architecture: Views and Beyond*. Addison Wesley, 2002.
- [8] Clements et al. *Evaluating Software Architectures: Methods and Case Studies*, Boston, MA: Addison-Wesley, 2002.
- [9] Dias, J. J. A. *Software Architecture Process for SOA-based Enterprise Applications*. MS.c dissertation, Federal University of Pernambuco, Recife, Brazil, August, 2008.
- [10] Dias, J. J.; et al. *A XML-based Quality Model for Web Services Certification*, In the 9th Int. Conf. on Enterprise Information Systems (ICEIS), Madeira, Portugal, 2007.
- [11] Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.
- [12] Erradi, A., Anand, S., Kulkarni, N. N. *SOAF: An Architectural Framework for Service Definition and Realization*, IEEE International Conference on Services Computing, 2006, 51-158.
- [13] Feuerlicht, G. and Lozina, J. *Understanding Service Reusability*, In the Proceedings of the 15th International Conference Systems Integration. Prague, Czech Republic, 2007, 144-150.
- [14] Gorton, I. *Essential Software Architecture*. Springer-Verlag Berlin Heidelberg, 2006.
- [15] Hofmeister, C., et al. *A general model of software architecture design derived from five industrial approaches*. The Journal of System and Software, 2007, 106-126.
- [16] ISO/IEC 42010. *Systems and software engineering - Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE Std 1471-2000. 1<sup>st</sup>. edition, 2007.
- [17] Jones, S., Mike, M. *A methodology for service architectures*, OASIS Draft, <http://www.oasis-open.org/>, 2005.
- [18] Kazman, R., Abowd, G., Bass, L. and Clements, P. *Scenario-Based Analysis of Software Architecture*, IEEE Software, Nov, 1996.
- [19] Kazman, R., Mark, R. and Clements, P. *ATAM: Method for Architecture Evaluation*, Technical Report. CMU/SEI-2000-TR-004. Pittsburgh, 2000.
- [20] Kruchten, P. *The 4+1 View Model of Architecture*. IEEE Software, vol. 12, no. 6, 1995, 45-50.
- [21] Matinlassi, M. and Kalaoja, J. *Requirements for Service Architecture Modeling*. In Workshop of Software Modeling Engineering of UML. Dresden, Germany, 2002.
- [22] Meyer, B. *Object-Oriented Software Construction*, 2nd edition, Prentice Hall, 1997.
- [23] Papazoglou, M. P. and Heuvel, W. *Service-oriented design and development methodology*. Int. J. Web Engineering and Technology, Vol. 2 No. 4, 2006, pp. 412-442.
- [24] Papazoglou, M. P. *What's in a Service?*. 1st European Conf. on Software Architecture. Proc. in Springer. Madrid, Spain, 2007.
- [25] Soni, D., Nord, R., Hofmeister, C. *Software Architecture in Industrial Applications*. In: Proc. of 17th International Conf. on Soft. Eng. (ICSE). ACM Press, 1995, 196-207.
- [26] Zimmermann, O., Krogdahl, P., Gee, C. *Elements of Service-Oriented Analysis and Design*, Available at: <http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/>, 2004.

<sup>1</sup> <http://www.ines.org.br>

# Research and Implementation of Service-Oriented Architecture Supporting Location-Based Services on Sensor Networks

Bin-Yi Liao<sup>1</sup>, Wen-Shyang Huang<sup>1</sup>, Jeng-Shyang Pan<sup>1</sup>, Hong-Chi Wu<sup>3</sup>, Yuh-Ming Cheng<sup>3</sup>,  
Jen-Kuin Lee<sup>4</sup>, Bo-Sian Wang<sup>3</sup>, E-Liang Chen<sup>3</sup> and Mong-Fong Horng<sup>1</sup>

Dept. of Electronic Department<sup>1</sup> and Dept. of Computer Sciences<sup>2</sup>

National Kaohsiung University of Applied Sciences, Taiwan

Dept. of Computer Sciences and Information Engineering Shu-Te University, Taiwan<sup>3</sup>

Dept. of Electronic Department, National Taipei University of Technology, Taiwan<sup>4</sup>

email: mfhong@cc.kuas.edu.tw

## Abstract

In this paper, a service-oriented architecture supporting location-based services on sensor networks is presented. Based on this architecture, an intelligent home service platform composed of software and hardware components are constructed to offer smart living functions and services. Based on SOA, the developed components communicate with each others through the interface of web services. Certainly, the system integration is also rapidly realized due to the interfaces of web services. A positioning health-care and service is presented through a case study. Consequently, the SOA approach is beneficial to fast composition of the target services. The realized system is helpful to improve the home living quality.

**Keywords:** SOA, Location-based services, Multimedia, Health-care service, Face recognition, Sensor networks,

## 1. Introduction

Digital home is a significant technology affecting deeply people living in 21 century. There are some critical technologies involved in digital homes; including information processing technology, information transportation technology and ICT hardware design and implementation technology. Through the technological integration, a location-based home service system is realized to provide health-care service, multimedia service and user identifying service. Among these software components, the interfaces of web services are introduced to construct a service-oriented architecture (SOA) platform. The benefits of SOA-based approach are (1) to offer highly feasibility for easy interconnection between software modules; (2) rapid customization for different user requirements; (3) open interface to connect Internet. As a result, the adopted architecture will be able to effectively collaborate with other web service systems. In this paper, we will present a SOA-based Location-Based Service (SLSS) platform. SLSS is composed of multimedia network, sensor networks, image processing, embedded systems and mobile devices to develop a service plug and play (SPNP) home environment. In this environment, home users enjoy (1) ubiquitous multimedia service on a hand-held device (2) location-based personalized service to enable specific services according the user location information detected by sensor

networks (3) user-identifying service based on face recognition to reliably identify home users. Due to the deployment with certain sensor nodes, a new ZigBee device compatible with IEEE 1451 to support XML messaging and software control. Thus, the developed devices efficiently assist the construction of the proposed SLSS. Consequently, the proposed SLSS is developed to prove the concept of a new living style, particularly in security, health and comfort, in future homes. The rest of this paper is organized as follows. Section 2 presents the related works to illustrate the technical background. The design and implementation of the proposed systems are presented in Section 3. The experimental results to verify the function and performance of the proposed SLSS are shown in Section 4. Finally, we conclude this in Section 5.

## 2. Related Works

As well known, a service-oriented architecture is essentially a collection of services. These services communicate with each other [1]. SOA is an architecture comprising (1) Loosely coupled services, (2) described by platform-agnostic interfaces (3) discovered and invoked dynamically (4) accessed in real-time (5) be transparent to users. Thus, a SOA application development is a collaborative effort from three parties: application, builders, service developers, and service brokers [2]. Additionally, Location-based service (LBS) is an application to provide personal and positional mobile service information according to user location. Users can access information and services related to the site at any time LBS has two basic functions, one is collecting spatial and positioning information, the other one is providing services according to user needs [3-4]. LBS [5-6] has been applied widely at this stage, especially on outdoor services, such as satellite navigation for automobiles, Google Map and Google Earth by Google, and PaPaGo and Urmapp e-maps. This study aimed to construct an integrated platform of LBS. The front-end positioning of this platform uses ZigBee location technology and room-based location technology for indoor location search. The information service provided by the backend information system is added to SPG through the Service Oriented Architecture (SOA) in Web Service mode, in order to realize the information flow mechanism of LBS. The introduction of SOA technique makes heterogenous systems easy to integrate.

Thus, in this work, a SOA platform is conducted on the basis of multimedia gateway, face recognition, ZigBee sensor networks, embedded system and hand-held devices. In the developed SOA platform, a service plug-and-play (SPNP) environment is presented to enable an intelligent home living. Certainly, typical applications are truly implemented to demonstrate the benefits of the proposal platform. Besides, in this intelligent home environment, various sensors are widely deployed. To solve the problem of connecting heterogeneous sensors, we will design and realize in the standard of IEEE1451. The smart ZigBee sensors compliant with IEEE 1451 offer the function of XML software control. Thus, the proposed SOA-compatible sensors will reduce the complexity of interfacing sensor with gateway server.

### 3. SOA Location-Based Service System (SLSS)

#### 3.1 System Architecture and Service Scenarios

The developed SLSS system is composed of four subsystems as shown in Fig. 1. The four subsystems are (1) SOA multimedia gateway of network and storage system (SMGNS); (2) Face Recognition Application and Control (FRAC); (3) ZigBee information Service (ZBS) and ZigBee Sensor Design for health-care Application (ZLA). SMGNS is designed as the home gateway connecting multimedia servers and storage systems to (1) offer multimedia content (2) store the information data from sensor networks. FRAC is a subsystem utilizing the technology of image processing to identify users. In FRAC, the user face is captured by a front-end camera. The face picture is analyzed to extract significant features such as shape and geometry information. Accordingly, the users are identified. The derived user identification will determine what services will be either turn on or turned off. ZBA is a Zigbee system to support the environment context sensing and relaying to the gateway. Thus, ZBA is an infrastructure of context-aware services in home. ZLA is a context-aware service system built on Zigbee network. The front-end of ZLA senses the signals of blood pressure; heart beat and body temperature as well as delivers the collected data to the back-end platform for further applications. The design details are illustrated as follows. SMGNS is a home gateway connecting Internet, appliances and sensor networks to construction a home network. Through SMGNS, home users access Internet multimedia services with high quality aided by effective QoS gateway [7-10].

SMGNS gives the following features to support the high-quality multimedia and home services;

- (1) QoS functions including service differentiation,
- (2) Remote appliance control
- (3) home automation
- (4) Load balancing

Since SMGNS has been realized by embedded system technology, the open and customized architecture benefit the integration, maintain and cost. FRAC is a user-identification application based on human face recognition. This application is also developed on an ARM-based embedded system. FRAC

utilizing Principle Component Analysis (PCA) [11-12] to extract the face features. The extracted features of human faces are categorized and maintained in a pattern base. The feature extraction is shown in Fig. 2.

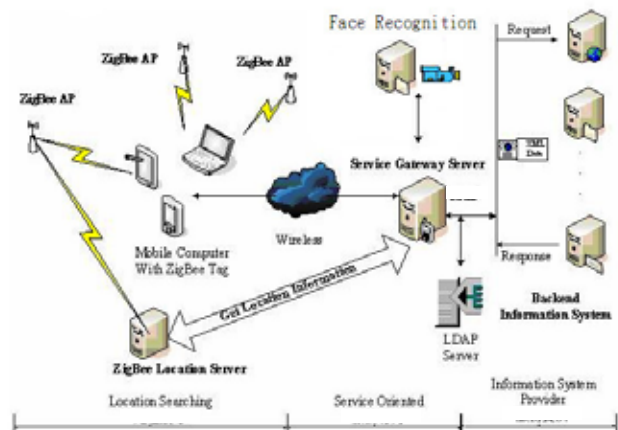


Fig. 1 SLSS system architecture

There are two kernel-based methods including Fisher Linear Discrimination (FLD) and Kernel Fisher Discrimination (KFD) to capture linear features and nonlinear features, respectively. Then a genetic programming approach is used to find the optimized category of pattern base. Once the face image is presented to be identified, the feature extraction is processed to obtain the features for matching. The pattern from pattern Use cases of the face-recognition application include house entrance control and authentication of appliance control.

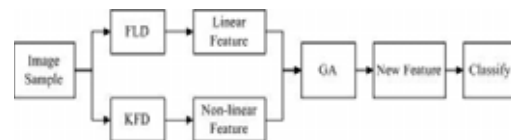


Fig. 2 Feature extraction of human faces

ZBS is a customized ZigBee sensor networks. A ZigBee sensor network is composed of two types of nodes; including sensor nodes and coordinator nodes. Sense node is responsible for the data sensing which is delivered through wireless channels to the coordinator. Each node is designed to relay the sensed data for its neighbors. The coordinator aggregates the sensing data from sensor nodes and forwards the context information to the backend monitor for further applications. There are two typical applications demonstrated in this works such as remote monitor and realtime notification. Remote monitoring enables the remote users to issue a request to the coordinator to query the sensing data. The sensor node will send the sensing data according to the request forwarded by the coordinator. Realtime notification is triggered by a trap caused by a pre-defined alert condition. Sensor nodes periodically acquire the context information. Once invalid context information is encountered, a trap is conducted to notify the invalid condition.

Based on ZBS, a smart ZigBee location-based application (ZLA) health-care service is built up In ZLA, there



are mobile sensor nodes to measure physiological data including blood sugar, blood pressure, heart beat. These collected data will be delivered to remote data servers on Internet, through SMGNS gateway. The service scenario is shown in Fig. 3. Each user wears a sense node to measure the target signals. The wearable sensor nodes communicate with the neighbor nodes to forward the measurement through multi-hop networks to the sensor dongle on the coordinator nodes. The operational steps of this service scenario are listed as follows,

- Step 1. The sensor node is triggered to measure the target physiological signals on a sensor node. Then the measurement is stored in the sensor node for further query.
- Step 2. SMGNS Gateway requests the measurement. The request is forwarded to the sensor network through dongle
- Step 3. The request is processed by the corresponding sensor node.
- Step 4. Data report is delivered to network routers.
- Step 5. The router forwards the data report to the dongle.
- Step 6. Gateway receives the requested data report.
- Step 7. Data report is uploaded to the data server for further applications.

In this scenario, the user location information is derivable from ZBS. Thus, wherever the user is, there is a route established between the sensor node and the coordinator. Along with the route, the physiological data is conveyed toward the coordinator in a manner of multi-hop relay. In other words, the data relaying is valuable to improve the overage of wireless sensor networks and to offer a low power consumption data transport service.

### 3.2 Service Architecture and Delivery

The presented SLSS is designed as a web service with XML messaging. The XML services, including multimedia services, health-care service, and other data services follow the web service architecture. XML web service is a service-oriented architecture to offer the functions (1) Service registration, (2) Service indexing and (3) Service delivery.

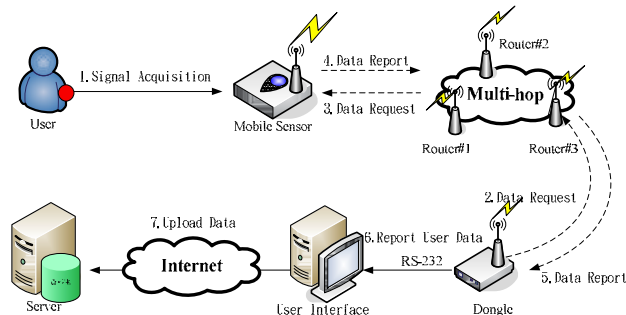


Fig. 3 Scenario of health-care services

Software components are encapsulated in the form of .NET framework. The messaging between clients and server follows Simple Object Access Protocol (SOAP) and eXtensible Markup Language (XML). At the client site, to access service

resource, in the front of user application, a service agent is designed as an interface to web server. This interface functions (1) Encapsulation and Decapsulation of service messages (2) access procedure for clients to obtain the service resource.

Typically, there are five parts to constitute web services follows,

1. Discovery: Accept the service registration from service providers and offer users with a list of service registration for service search/discovery
2. Description: Prompt the description of accessible services including usage illustration and example for users.
3. Messaging: Define the message format and handshaking procedure for client and server to correctly exchange messages and service contents.
4. Encoding: Provide a consistent encoding/decoding to ensure message validation between client and server.
5. Transport: Convey the encoded and encapsulation service content from server to the requesting client via specific transport protocols such as TCP/IP.

SLSS is developed according to the following steps:

1. On the backend platform, the service registration, service indexing and service locating are implemented according to UDDI (Universal Description, Discovery and Integration)
2. Registered service is presented according to WSDL(Web services Description Language). WSDL is an XML-based description language to illustrate the methods, interface and return value of web services.
3. SOAP is in charge of the communication format between the located server and backend platform. By use of SOAP, the agent at clients knows what service is and how to retrieve service in a defined message format.
4. All messages exchanged between servers and clients are in a XML format designed for the specific applications.
5. Hyper Text Transport Protocol is the standard used by clients and servers to transport messages and service contents on networks.

Thus, through the above steps, the developed services are registered as a web service with its method and interfacing on the backend platform. The backend platform is service repository and implements a friendly management console.

### 4. Experimental Results

A prototype of SLSS has been realized to prove the operational concepts presented in the previous sections and to present a series of location-based services such as multimedia, security and health-care services. The specifications of the SLSS system and four subsystems are listed in Table 1. Embedded systems are the dominant approach to establish the SOA platform because it is advantageous in customization of hardware/software design and cost reduction. The software tools of system development are open sources to enhance the reusability of software components. Besides, the interconnections between subsystems are based on open

system interconnection, such as IEEE 1451, TCP/IP and XML. There are four LBS implemented in the prototype; multimedia streaming, face-recognition entrance control, home appliance control and remote physiological signal measurement. Due to page limitation, the function and performance evaluations of multimedia streaming service and remote blood pressure measurement are demonstrated.

Table 1 SLSS system specification

SOA Location-Based Service System				
Subsystem	SMGNS	FRAC	ZBS	ZLA
Hardware Spec.	ARM2410, WiFi/Fast Ethernet	Xscale 270/WiFi/Web Cam /GPIO	MCU /ZigBee MAC/RF/WiFi	MCU /ZigBee MAC/RF
Software Develop. tools	Embedded Linux, C, PHP, MySQL	Embedded Linux, C,	C	C
Functions	QoS Gateway/ Storage server	Face-recognition/Security/ Appliance Control	Context data Convey/ Context Monitor.	Physiological signal sensing.
Protocols/ Standards	DiffServ/ TCP/IP/ XML	PCA/FLD/KLD /GA/TCP/IP /XML	ZigBee/TCP/ IP /XML	IEEE1451/ ZigBee/ TCP/IP



Fig. 4 Comparison of multimedia streaming performance with (a) /without (b) QoS guarantee

First, the performance of multimedia streaming service quality with no QoS guarantee is shown in Fig. 6. In traditional IP networks, the multimedia stream could be corrupted by the bandwidth insufficiency or instability. There will be packets lost when the bandwidth is insufficient. Unfortunately, packet loss causes frame corruption as depicted in Fig. 4(b). In contrast, when the bandwidth guarantee is enabled to guarantee the stream bandwidth, the service quality is improved as shown in Fig. 4(a)

## 5. Conclusions

In this paper, a prototype of SOA Location-based Service System is presented. The SLSS system is designed to be a SOA application on intelligent home living. SLSS makes uses of wire/wireless networks, Sensor networks, face image processing, and pattern recognition to construct various services, including multimedia streaming services, user-identifying service, and health-care service, for home users. The locations of users requesting services are available from the designed sensor network. According to the user locations, a certain service with proper quality is accessible by the users. The locating, indexing and delivery of service content are collaborated by the four subsystems of SLSS. To glue the four subsystems, a SOA protocol stack, including UDDI, WDSL,

SOAP and XML, are developed to facilitate the communication between subsystems. The experimental results indicate the present SLSS is beneficial to smart living for home users. Also the SOA approach is proved its feasibility and high efficiency to compose a service from available service components.

## Acknowledgement

The authors would like to thank the partial financial support from National Science Council, Taiwan under the grant 96-2218-E-151-002.

## References

- [1] E Newcomer, G Lomow, "Understanding SOA with Web Services," Addison-Wesley Professional, 2004.
- [2] W.T. Tsai, Y. Chen, R. Paul, "Specification- Based Verification and Validation of Web Services and Service-Oriented Operating Systems", Tenth IEEE International Workshop on Object-oriented Realtime Dependable Systems (WORDS 05), Sedona, February 2005, pp.139 - 147.
- [3] Sayed Hashimi, ".Service-Oriented Architecture Explained," available at <http://www.ondotnet.com>
- [4] Todd Datz, "What You Need to Know About Service-Oriented Architecture," available at <http://www.cio.com>
- [5] P. Ordóñez, P. Kodeswaran, V. Korolev, W. Li, O. Walavalkar, B. Elgamil, A. Joshi, T. Finin, Y. Yesha, and I. George, "A Ubiquitous Context- Aware Environment for Surgical Training", Conference on Mobile and Ubiquitous Systems: Networking & Services, Aug. 2007, pp. 1-6
- [6] Y. Xia, and H. Y. Bae, "General Platform of Location based Services in Ubiquitous Environment", International Conference on Multimedia and Ubiquitous Engineering, April 2007, pp. 791-795.
- [7] Takeshi Saito, Ichiro Tomoda, Yoshiaki Takabatake, Junko Ami and Keiichi Teramoto, "Home gateway architecture and its implementation", IEEE Transactions on Consumer Electronics, Vol. 46, No.4, pp. 1161- 1166, November 2000.
- [8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF RFC 2475, December 1998.
- [9] D. Grossman, "New Terminology and Clarifications for Diffserv," IETF RFC 3260, April 2002.
- [10] Malik Khan, "Quality of service can also deliver performance monitoring," May 2000, <http://www.convergedaccess.com/>
- [11] Chengjun Liu, Gabor-Based Kernel PCA with Fractional Power Polynomial Models for Face Recognition, IEEE Trans. Pattern Ana. Machine Intell., Vol. 26, No. 5, May, 2004, pp 572-581.
- [12] P. N. Belhumeur, J. P. Hespanha, D.J. Kriegman, Eigenfaces vs. Fisherfaces : Recognition Using Class Specific Linear Projection, IEEE Trans. Pattern Ana. Machine Intell., Vol. 19, No. 7, July, 1997, pp 711-720.

# Service Creation and Composition for Realization On Service-Oriented Architecture

Chi-Lu Yang<sup>1,2</sup>, Yeim-Kuan Chang<sup>1</sup>, Chih-Ping Chu<sup>1</sup>

<sup>1</sup>Department of Computer Science and Information Engineering, National Cheng Kung University

<sup>2</sup>Innovative DigiTech-Enabled Applications & Service Institute, Institute for Information Industry

<sup>1</sup>Tainan, Taiwan R.O.C.

<sup>2</sup>Kaohsiung, Taiwan R.O.C.

stwin@iii.org.tw, ykchang@mail.ncku.edu.tw, chucp@csie.ncku.edu.tw

***Abstract** - Applying ICT to assist daily activities and interests is already a worldwide trend. Through software and network techniques, computers could remotely provide various services. Up to now, Service-Oriented Architecture (SOA) has become one of the most popular techniques to realize these daily services in various areas. Therefore, we would need systematic methods for services creation from a service-centric viewpoint. In this paper, we first illustrated the progress of service realization. The interoperability of service creation and realization along its life cycle were carefully explained. With methodical sequences, domain-specific services can be well created and designed. The home-care services were demonstrated in the case study. These services were verified by a pilot trial in a real environment. The experimental results showed the usability and effectiveness. We believe that a successful experience on service realization is worthy to be shared.*

**Keywords:** SOA design, service creation, service composition, service verification, digital home-care

## 1. Introduction

Using Information and Communication Technologies (ICT) to create innovative services models, many enterprises have continuously become famous worldwide, such as YouTube, WRETCH, Google, Amazon, among many others. Their models have successfully provided services to promote business values over the Internet. One of their common features is that they had applied specific information techniques with the network techniques.

Applying ICT to assist daily activities and interests, such as medical care, lifestyle, traffic, education, and entertainment is already a worldwide trend. Through software and network techniques, computers could remotely provide various services. Up to now, Service-Oriented Architecture (SOA) has become one of the most important techniques to realize these daily services in various areas. For example, the SOA-based system was proven to effectively enhance the homecare services [5], [11]. With SOA, various stakeholders could

be linked into similar service processes. Moreover, these processes are closely conjoined with their services.

Therefore, we would like to determine how/what services would be realized by SOA techniques. The solutions are inquired from a service-centric viewpoint. In this paper, we illustrate the progress of service realization and share our experiences from a case study at the same time. In Section 2, the SOA principles and challenges are introduced. In Section 3, the interoperability of service creation and realization are proposed along its life cycle. The representations of service elements are even illustrated, since we need large communication with various providers during service creation. A case study on realizing home-care services is carefully explained in Section 4. A pilot trial of the case study is set up for verification. The experiments and their results are discussed in Section 5. The summary of the study is then given in Section 6.

## 2. Related Work

Service-Oriented Architecture (SOA) is a software architectural style for realizing and constructing business services, which are composed by components as services [1]-[3]. Service-oriented technology could expand information and communication technology (ICT) to provide various services, which frequently require a large amount of data exchange. SOA also separates services into distinct units such as components or modules, which can be deployed over the Internet and can be reused to compose new applications. By SOA, services can be delivered to end-users over the Internet.

The architecture of SOA is clearly layered out. Business services could thus be clearly identified and layered in SOA. Business services are also created and composed by various software components by SOA. The typical layers of SOA are business process layer, service and application layer, and technical layer [4]-[5].

Furthermore, the general architectural principles figure out the ground rules of SOA for development, deployment, and maintenance [4]-[5]. The four ground rules in this study are service modeling, deliverability, compliance to standards and reusability. These are described as follows:

- Service modeling - Service definition and creation, deployment and delivery, monitoring and tracking, service concept, key performance indicators (KPI) definition, and so on.
- Deliverability - A service on SOA should be delivered via the Internet. The charged fee would be accounted for by the service providers.
- Compliance to standards – A large number of messages is frequently exchanged through the SOA platforms. These exchanged messages will extend the SOA capability and result in significant issues for standardization, identification, authorization, security, privacy, and so on.
- Reusability - A segment of the service might be reusable to compose new services. In other words, components or modules in SOA would be reused in various business processes and even mobile services.

In addition, the specific principles for service design and creation are categorized into two types. The first type includes specific design guidelines of SOA for service providers. The second type deals with the interaction between the service providers and consumers. They are described as follows:

- Service abstraction - Services are logically hidden from the outside world, beyond what is described in the service contract.
- Service autonomy - Services have control over the business processes they encapsulated.
- Service encapsulation - Various services in the Internet are consolidated with Web services under the SOA platform.
- Service composition - Collections of units of services can be coordinated and combined to create services.
- Service discoverability - Services are designed to be accessible to the public, they can thus be found and accessed via available discovery mechanisms.
- Service loose coupling - Services maintain a relationship that minimizes dependencies on one another.
- Service optimization - High-quality services are generally considered more than low-quality ones.
- Service contract - Services are attached to the communicable agreements, and are defined in service description documents.

Building SOA is not only a technical challenge, but also a business challenge. In the visions of SOA, relationships between service consumers and providers are not tightly stipulated. Their relations are loose coupling [6]. Thus, consumer services are not forcefully influenced by the changes made by the providers. Moreover, consumer service interacts with the service provider based on the service contract. Thus, negotiating Service Level Agreements (SLA) is even a critical issue. The SLA should even satisfy some general and specific principles.

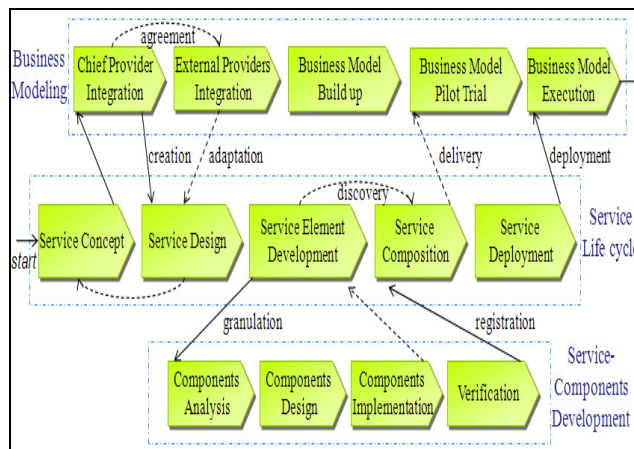


Fig. 1 Service Development Life Cycle

Another constraint is that SOA applications almost have to be used in a distributed environment [7]-[8]. That means end-users and service providers are distributed geographically. Services in SOA are delivered via the Internet. Unfortunately, SOA provides an environment that is convenient for hackers and intruders [9]-[10]. Web service is one of the most important ways to implement SOA. The relevant techniques are Extensible Markup Language (XML) [13], Web Services Description Language (WSDL) [14], Simple Object Access Protocol (SOAP) [15], and Universal Description, Discovery and Integration (UDDI) [16].

### 3. Service Creation

Service creation is a critical start when we would like to provide services by SOA. A service is composed of various elements, which are provided by different providers. These providers are even geographically distributed. If they are connected with the SOA platform, they will have chance to cooperatively create new services. In order to know how a service is realized, the life cycle of a service will be explored along business modeling and component development in the following sub-section. The representation of service elements will be also introduced.

#### 3.1 Service Development Life Cycle

A service is intensively related with its business model since the service will be realized for serving someone/something in the real world. The life cycle of services will approach business modeling. These phases are defined as service concept, service design, service element development, service composition, and service deployment. The development life cycle of service is shown in Fig. 1. Furthermore, the service elements are closely associated with software lifecycles.

The service concept is initiated by interviewing with domain experts. Its draft will be used for triggering internal integration of chief provider, who is the key provider in the service concept.

The service design, which is constructed by service designer, will show service scenarios step by step. Each

step corresponds to a service element. The service element would therefore possibly be reused. The service scenarios are extended based on the service concept and are adapted by external providers. These providers will be included and organized if the scenario requires them to provide specific service elements. The chief and external providers cooperate by following service level agreements (SLA). The agreements should at least include the service elements and charge flow. The regular routines among these providers will be extremely large after the business model is executed in a real environment. Therefore, they need software and internet to manage these routines.

Service elements are granularly developed to be components, which are even encapsulated to web services and are composed to provide services on the Internet. The service flow and charge flow are hidden in these components. A tool engine for service management is also indeed essential.

Since these derived components are analyzed and designed by object-oriented methods, they could characteristically be reused to compose new services. Verification and registration of the components are necessary before they are discovered and composed. The interfaces are the most important during service composition. It's better to standardize interfaces.

Before these services are actually deployed and executed upon the business model, they have to undergo a pilot trial in a real environment. During the pilot trial, some problems will occur. The providers, especially the chief provider, must solve these issues for smoothing future business operation.

### 3.2 Service Element Representation

A service, which is abstract in the services concept, would be expanded into a service scenario step by step. A service is composed of service elements. The types of service elements include services, roles, proprietors, devices, transactions, and locations. A service here is represented as the following figure.



A role in the services stands for an individual who performs a specific task. The role frequently has professional knowledge and plays an important part in this service. For example, the defined roles in home-care services are doctor, nurse, centre-staff, care-giver, and so on. A role in the services concept is represented by a circle with a person within it, such as the following figure.



A proprietor in the services is responsible for providing general service. This could be combined and replaced into the service through SLA. The collaborative policies are derived from proprietors. For example, the proprietors are the drug deliverer, the hospital, the transporter, and the ambulance, among others. A proprietor is represented by a circle with a service provider such as a car or a building within it. The representation is shown as the following figure.



A device in the services indicates a technical system which could be software, hardware, end-device, system or platform. The technical systems are created through ICT, which are especially focused on software techniques and communication techniques. For example, the defined devices are the homebox, service platform, and bio-signal measurable devices. The homebox, which was developed in our project, is a gateway for collecting and transferring personal bio-signals in the patient's house. Service platform is a software platform for service integration, delivery, and management. An example of the device representation is shown as the below figure.



A transaction is represented as a solid line with a bi-directional arrow. An action(s) on the line is (are) an executable action(s) between a service and a role (or a proprietor). An action would be separated into sequential items in the service scenario. The representation of a transaction is shown as the following figure.



A location is a position where a person or a device is located. Delivering services to a remote side is an important action in modern business models. The services are provided via a communicable network because the service providers and consumers are located in different positions; thus a demarcation of the location is necessary. For example, the defined locations are home, care center, and hospital. An example of location representation is shown as the following figure.



Services are controlled and monitored by a specific role. A dotted line between a service and the role means that the service is uniformly monitored by that role. The progress of a service would be monitored if it is necessary. A service should have features of reliability

and efficiency in a real-time system. The services in real life are frequently complex; we would indeed have to manage them using a well-defined service platform, such as service-oriented platform.

#### 4. Case Study

In the home-care services area, patients with chronic diseases need long-term care at home. If they are hospitalized for a long time, a lot of costs are entailed, such as the financial and emotional burden on their families as well as wastage in hospital resources. Therefore, one effective solution is to remotely take care of patients at home. However, that would be a challenge for both the patient's family and the hospital. In this case study, the home-care services are developed by following service life cycle. The service elements introduced in the previous section are also included in the home-care services concept. The services concept in chronic home-care area is shown in Fig. 2.

The home-care services are derived from the services concept. By composing the service elements, service scenarios could be created. Two service scenarios, named as health status monitor and emergency medical treatment, are described in the following sub-sections.

##### 4.1 Health Status Monitor

The patient's health status is regularly monitored by homebox and bio-signal equipment in the house. This service is triggered three times every day. First, homebox sends bio-signals to the SOA-based healthcare platform. When the platform receives the signals, these will be automatically judged by inferable components, which are pre-installed into the platform. If unusual signals are detected, an alert for the patient will be sent to centre-staff's monitor. At the same time, the staff can obtain the patient's conditions through a telephone call, if it's connected. At the same time, the patient's EHR and conditions will be sent to doctor's computer through the service platform for getting the doctor's recommendation. The centre-staff can then quickly take care of the patient and process exact actions for him/her. The service scenario of the health status monitor is shown in Fig. 3.

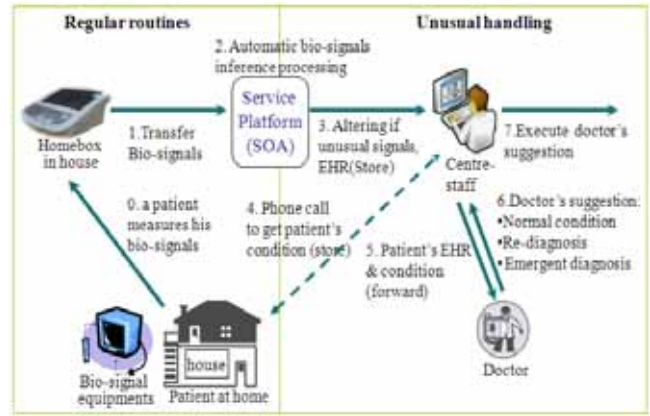


Fig. 3 Health Status Monitor

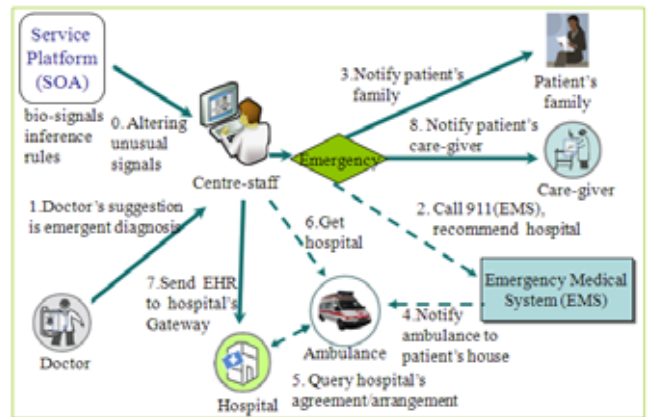


Fig. 4 Emergency Medical Treatment

##### 4.2 Emergency Medical Treatment

The service for patient's emergency care is designed on the service platform. The service will be triggered by the unusual bio-signals, and while the doctor also suggests the patient to diagnose emergently. Thus the centre-staff will immediately call the emergency medical system (EMS). At the same time, the SOA-based healthcare platform is notifying patient's family. While the patient is being taken to the hospital by an ambulance, his/her EHR and the unusual signals are also sent to the hospital (HIS system) through the specific gateway in the service platform. Finally, the patient's emergency will be sent to his/her care-giver PDA. The service scenario of emergency medical treatment is shown in Fig. 4.

The SOA-based healthcare platform is layered out and designed by following SOA principles. Its architecture is in the reference [5]. The service platform can provide executable environments which support standardized messages, various interfaces and flexible connections. Different services techniques and providers could cooperate on this platform. Those messages among providers are passed through a specific message gateway [11]-[12]. The software components are derived from these service elements and are implemented to build up the platform. The components are in described by following Unified Modeling Language (UML) and programmed by C#.NET language.

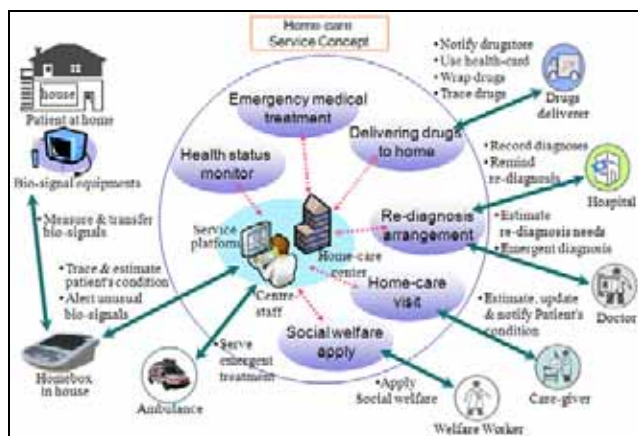


Fig. 2 Home-care Services Concept

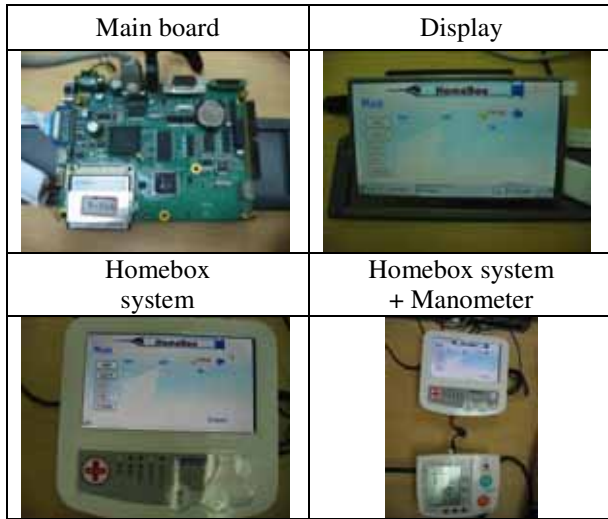


Fig. 5 Homebox system

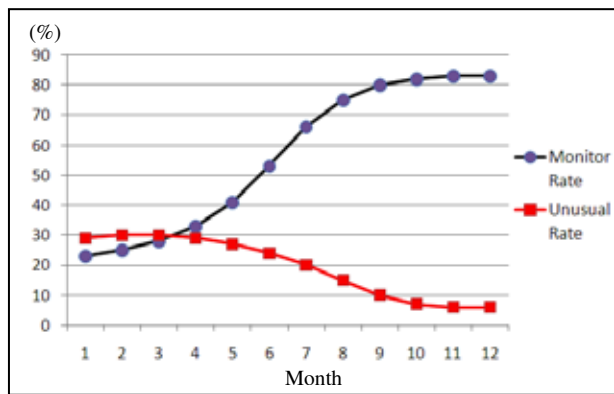


Fig. 6 Rating Services

## 5. Experiment – A Pilot Trial

### 5.1 Experiment Environment

The homebox is an embedded system developed by our team in 2006. It is a message gateway used for collecting and transferring patient's bio-signals with his/her conditions to a remote server. At the local site, it could connect bio-signal equipment using RS232 and USB2.0 interfaces. An Ethernet port is also built in the box. Its kernel core is PXA270 with 312MHz CPU and 64MB SDRAM. A 512MB SD card and boot ROM are used to boot the system. Linux kernel version 2.6 is pre-installed as its operation system. A 7" Widescreen LCD with 480x234 Resolution is used for its display. Four user defined hot keys on its surface are used for pressing *Yes*, *No*, *Enter*, and *Display on/off*. The homebox is one of the main devices connected to enable healthcare services at the client site. The homebox is shown in Fig. 5.

With our specific components, the homebox installed in patient's house is reliably on-line for 24 hours a day. The message gateway is even pre-installed to remotely call web services. All components are implemented by C#.NET language. One hundred homeboxes were set up in 100 patient's houses in Kaohsiung City, Taiwan.

The SOA-based healthcare platform is set up in the

home-care service center, where the staffs are. A tool for controlling service flows is applied in the service platform. With the tool, centre-staff can be aware of services' progresses. The customized components in the platform are developed by C#.NET language. The users' profiles and bio-signals are permanently stored in MS SQL server 2005. The services scenarios mentioned in Section 4 are experimented in a real environment. One hundred chronic patents were involved in the trial. The pilot trial lasted from September 2007 to August 2008 in Kaohsiung City. The chief medical provider is Chung-Ho Memorial Hospital.

### 5.2 Experimental Results

The usability of the health status monitor service was rated during the pilot trial. The usage rate was below 30% during the first three months. After eight months, the usage rate increased and reached 80%. This means that more patients got used to the service in the long run. The usage rate was stable up to 80% in the last three months. Patients treat using the service as their daily job. When their bio-signals are forwarded and stored in the back-end server, their health statuses are regularly monitored.

The patients' unusual rate was close to 30% before they joined the pilot trial. However, they cannot be sensibly taken to the hospital when they encounter an unusual condition. The average time of taking them to the hospital is between 10 to 15 hours. However, the medical treatment for them is effective within three hours. Fortunately, our emergent service of medical treatment is enabled when unusual condition is detected. This service could take a patient to the hospital within an average of 35 minutes. Therefore, the variation that a patient will be saved in a critical condition is fairly significant.

The other services were also validated by the pilot trial in this study. The care-givers visited patient's home and took care of them twice a week. They evaluated and recorded the patients' health and life skills. That was why the unusual rate was reduced to below 10%. The rated services are shown in Fig. 6.

## 6. Conclusion

Up to now, Service-Oriented Architecture (SOA) has become one of the most important techniques to realize those daily services in various areas. Therefore, we should explore systematic methods to realize services from the service-centric viewpoint. In this paper, we carefully explained the interoperability of service creation and realization along its life cycle. Using the methodical sequences, domain-specific services can be well created and designed. The representations of service elements were even used for communicating among providers during service creation. We had validated our methods by the pilot trial in the home-care case study. The experimental results had represented its usability and effectiveness. We believe that a successful experience on service realization is worthy to be shared.

## 7. Acknowledgements

This research was supported by the 2th Applied Information Services Development and Integration project of the Institute for Information Industry (III) and sponsored by MOEA, Taiwan R.O.C.

Care-experts and pilot patients were supported by the Department of Medical Information, Chung-Ho Memorial Hospital in Kaohsiung, Taiwan R.O.C.

## 8. References

- [1] Thomas Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, 2004.
- [2] Dave Hornford, *Definition of SOA*, The Open Group, October, 2006.
- [3] Choudhary, V., "Software as a Service: implications for investment in software development," in *Proceedings of the 40th Annual Hawaii International Conference on System Science (HICSS'07)*, Los Alamitos, CA, USA, 2007, pp. 209-218.
- [4] Yvonne Balzer, "Improve your SOA project plans," IBM Global Services, July 2004.
- [5] Chi-Lu Yang, Yeim-Kuan Chang and Chih-Ping Chu, "Modeling Services to Construct Service-Oriented Healthcare Architecture for Digital Home-Care Business," in *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'08)*, San Francisco, USA, July, 2008.
- [6] Eric Newcomer and Greg Lomow, *Understanding SOA with Web Services*, Addison Wesley, January, 2005.
- [7] Asit Dan, Robert Johnson and Ali Arsanjani, "Information as a service: modeling and realization," in *Proceedings of International Workshop on Systems Development in SOA environments (SDSOA'07)*, IEEE Computer Society, Los Alamitos, CA, USA, May, 2007.
- [8] Gennaro Cuomo, "IBM SOA on the Edge," *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ACM Press, New York, NY, USA, 2005, pp 840-843.
- [9] Ned Chapin, "Service Granularity Effects in SOA", in *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'08)*, San Francisco, USA, July, 2008.
- [10] H. Xu, M. Ayachit and A. Reddyreddy, "Formal Modeling and Analysis of XML Firewall for Service Oriented Systems," *International Journal of Security and Networks (IJSN)*, Vol. 3, No. 3, 2008.
- [11] Chi-Lu Yang, Yeim-Kuan Chang and Chih-Ping Chu, "A Gateway Design for Message Passing on SOA Healthcare Platform," in *Proceedings of the 4th IEEE International Symposium on Service-oriented System Engineering (SOSE'08)*, Jhongli, Taiwan, Dec., 2008.
- [12] Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker and Ronald Monzillo, *Web Services Security: SOAP Message Security 1.0*, Organization for the Advancement of Structured Information Standards (OASIS), March, 2004.
- [13] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau eds. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, World Wide Web Consortium (W3C) Recommendation, Nov., 2008.
- [14] David B., Canyang Kevin L., Roberto C., Jean-Jacques M., Arthur R., Sanjiva W. et al. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C: <http://www.w3.org/TR/wsdl20-primer/>, 26 June 2007.
- [15] Martin G., Marc H., Noah M., Jean-Jacques M., Henrik F., Anish K., Yves L. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, W3C: <http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>, 27 April 2007.
- [16] Tom B., Luc C. and Steve C. et al. *UDDI Version 3.0.2*. OASIS: <http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf>, 19 October 2004.



# An Extendible Translation of BPEL to a Machine-verifiable Model

John C. Sloan, Taghi M. Khoshgoftaar, Augusto Varas

Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Florida USA  
taghi@cse.fau.edu

## Abstract

*Model checking can exhaustively verify if a Business Process Execution Language (BPEL) program correctly orchestrates activities amongst a collection of web services. We automate construction of a machine verifiable model given a BPEL program and a set of modeling assumptions. In particular, we define an intermediate representation that is both extendible and supports rule-based generation of these models. This representation captures the notion of context in terms of a finite state transducer. Context enables an analyst to express and confine modeling assumptions to specific basic or structured activities inside a BPEL artifact. Finally, we present a subset of production rules for converting the intermediate representation into a model in Promela – the language used by the Spin model checker. We illustrate these ideas with an excerpt from a case study.*

**Keywords:** web services, BPEL, formal verification, model checking, Promela, Spin.

## I. Introduction

*a) Motivation:* Orchestrating web services involves combining loosely coupled autonomous services, each having its own interface, advertised functionality, and specified behavior. Such orchestration of black-box services frees developers from low-level concerns involving platform, implementation, and versioning.

These freedoms render white-box testing techniques ineffective and, for safety or fiscally critical systems, suggest the need for exhaustive verification techniques like model checking. An orchestration will behave unexpectedly if it is subject to implicit assumptions. Explicitly representing these assumptions in a machine-verifiable way will help safeguard against compositions that exhibit undesirable forms of service interaction.

Recent efforts have been made to automate the generation of machine verifiable models from web service artifacts. Such automation seeks to minimize both human effort and judgment in model capture. These efforts, while laudable, have resulted in modeling artifacts that are difficult to understand, troubleshoot, and extend.

*b) Contributions:* This paper outlines an extendible approach to automating the translation of BPEL source code to a machine-verifiable target model. It improves on earlier approaches, by enabling the practitioner to specify assumptions intended for only *portions* of the target model. Portions of the composition that have been deemed reliable can be abstracted to models having smaller state spaces, while newer less reliable portions can be modeled in greater detail.

*c) Overview:* A verifiable model must capture what is both *stated* and *implied* by the BPEL artifact. We will define a translation of a BPEL artifact (i.e., what it *states*) into a relation, defining and applying notions of *frame*, *slot*, and *context* to the translation process. In particular, this relation will represent a collection of lexically scoped *frames*, with each frame comprised of a collection of tuples known as *slots*. We make assumptions about the behavior *implied* by the BPEL execution model, by adding slots into certain frames. In the case study, we added modeling assumptions concerning atomicity and synchrony. Space did not permit us to discuss assumptions that enable the verification of more pessimistic models that feature non-determinism.

Computing the value of a *context* entry for each slot enables us to specify the scope and effect of slots that represent assumptions. We construct a finite state transducer to compute this context along with the three other *content-related* entries that comprise each slot.

Given a representation enriched by a number of assumptions, we formulate a set of rules for translating each frame into an expression in the target modeling language. Each rule is comprised of atomic propositions, each referencing a portion of certain slots. As a Boolean expression over these atomic propositions, a satisfied antecedent will generate an expression in the target modeling language.

We provide the needed intuition with an excerpt from a case study. Based on a prototype under development, we lend practical insights into how this approach is being implemented.

The rest of this paper is organized as follows: Section II describes the translation from a BPEL artifact to

frames. Section III presents rules for translating frames to expressions in the target modeling language Promela, while discussing the impact of added assumptions. Section IV examines work related to automated translation of BPEL to verifiable models. Finally, Section V provides a summary and brief description of future work.

## II. BPEL to Frames

A BPEL artifact can be regarded as a collection of nested XML elements forming some tree structure. A translation must preserve its meaning and nested structure while simplifying its form. Conceptually, this involves squeezing the *syntactic sugar* out of the BPEL artifact, leaving only a single four column table that captures what the artifact expressly *states*. We then add entries to the tabular form to capture what is *implied*, particularly assumptions about how BPEL is executed. As a running example, we used an abbreviated version<sup>1</sup> of the BPEL artifact for the Purchase Order Process appearing in the WS-BPEL Specification [1]. This section defines the translation into a tabular form that preserves both context and content.

### A. Computation of context

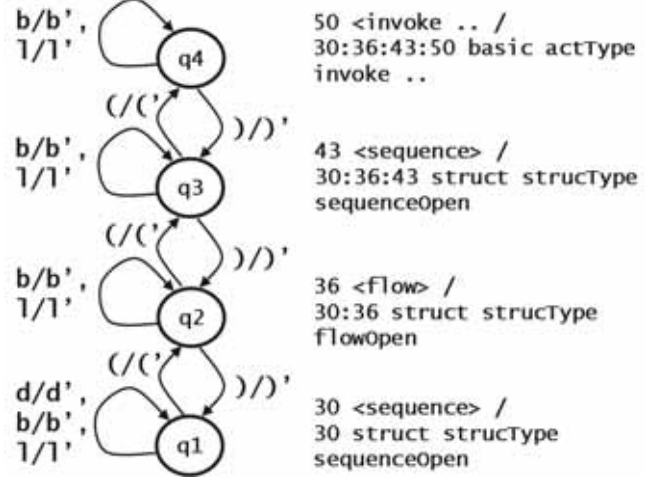
Computing the context of any element in a BPEL artifact, enables us to confine modeling assumptions to some non-global process scope. We describe this computation in terms of deterministic finite state transducer  $M = (Q, q_0, q_f, \Sigma, \Gamma, \delta, \phi)$ . For the set of states  $Q$ , both initial state  $q_0$ , and final state  $q_f$  are the same. Given alphabets  $\Sigma$  and  $\Gamma$ , which we describe in the next paragraph, we define state-transition function  $\delta : Q \times \Sigma \rightarrow Q$ , and output function  $\phi : Q \times \Sigma \rightarrow \Gamma$  in a way suggested by Figure 1. Central to this construction is how we define input and output alphabets  $\Sigma$  and  $\Gamma$ .

We abstract the structure and relationships between BPEL's basic activities, structured activities, and control links using input alphabet  $\Sigma$ . Each  $\sigma \in \Sigma$  can represent an entire BPEL element as in the case of declarations, links, and basic activities. Each  $\sigma$  can also represent either the opening or the closing portions of a structured activity. The left hand side of Table I lists each input symbol, a brief description, and an example.

Output alphabet  $\Gamma$  bears a 1-to-1 correspondence to  $\Sigma$  for each symbol  $\gamma_i \in \Gamma$  retaining the same meaning as its corresponding symbol  $\sigma_i \in \Sigma$ , changing only its form. The right hand side of Table I lists these output symbols and examples of their output forms. These alphabets will enable us to preserve the nested structure by using a finite depth stack.

The end result is the automaton in Figure 1. The left-hand side of Figure 1 shows a finite state transducer that

translates a BPEL artifact into a sequence of four-tuples, an example of which appears to its right. Defining this abstracted alphabet enabled us to construct a finite state machine that can process nested elements using a stack of *fixed* maximum height. In the interest of space, Figure 1 does not show state  $q_0$  which would have appeared below state  $q_1$ . State  $q_0$  is both initial and final and represents the bottom of the stack.



**Fig. 1. Finite state transducer  $M$ , and sample input/output**

As the example on the right-hand side of Figure 1 shows, only the top of the stack can process an element for either a basic activity  $b$  or a control link declaration  $l$ . As each opening portion ( of a structured activity is encountered, its location gets pushed onto the stack, modeled by the transition from state  $q_i$  to  $q_{i+1}$ . Conversely, each closing portion ) pops the stack. From the example on the right-hand side of this figure, one can discern the value of context stack 30 : 36 : 43 : 50 which will comprise the first of four entries in some slot.

### B. Computation of content

To discuss the remaining three entries, all of which relate to content, we must first define the notion of *frame*. A frame is a collection of all four-tuples (or *slots*) that in some way share the same value of context stack  $c$ . Examples of a frame include the collection of all slots that represent some declaration or basic activity. Furthermore, a frame may include slots for all basic and structured activities nested within a single structured activity or process scope. More formally, frame  $\mathcal{F} \subseteq R$  is a subset of some relation  $R \subseteq C \times T \times A \times V$  in which some context  $c \in C$  is subject to the following requirement. For some context stack value  $c_i \in C$  and prefix  $c_{\leq i} \preceq c_i$ , all tuples in  $\mathcal{F}$  share the same common prefix, namely  $c_{\leq i}$ . Since the ordering of  $R$  is the same as the ordering

<sup>1</sup>Download from: <http://www.osoa.org/display/Main/Relationship+between+SCA+and+BPEL>

$\sigma$ :	Description:	Examples:	$\gamma$ :	Examples:
$d$	declarations	<code>&lt;partnerLink name="pOP" .., &lt;variable name="vPO" .. /&gt;</code>	$d'$	01:13 partner plName pOP 01:24 variable vName vPO
(	structured activity begin	<code>&lt;flow&gt; &lt;sequence&gt;</code>	(	01:30:36 struct strucType flowOpen 01:30:36:43 struct strucType sequenceOpen
$l$	control links	<code>&lt;link name="xStI" .. /&gt;</code>	$l'$	01:30:36:40 link lDecl xStI
$b$	basic activities	<code>&lt;invoke partnerLink="pWhs" ..&gt;..&lt;/invoke&gt;</code>	$b'$	01:30:36:43:50 basic pLink pWhs 01:30:36:43:50 ..
)	structured activity end	<code>&lt;/sequence&gt; &lt;/flow&gt;</code>	)'	01:30:36:43 struct strucType sequenceClose 01:30:36 struct strucType flowClose

TABLE I. Input  $\Sigma$  and output  $\Gamma$  alphabets for Transducer  $M$

of its BPEL artifact, by construction of machine  $M$ , all such slots will be grouped together into one equivalence class, namely frame  $\mathcal{F}$ . An example of this is shown in the `<sequence>` construct starting at line 43 in Table II. Here, 01:30:36:43 represents prefix  $c_{\leq i}$ , while 01:30:36:43.44 may represent some  $c_i$ . Also note, predecessor  $c_{\leq i-1}$  of  $c_{\leq i}$  is 01:30:36:41 and successor  $c_{\leq i+1}$  is 01:30:36:59.

The second entry in the slot, and first content-related entry, is a text identifier corresponding to a symbol in the output alphabet. It denotes the type  $T$  of some frame  $\mathcal{F}$ . For basic activities and declarations the third and fourth entries denote attribute  $A$  and value  $V$  corresponding to some attribute-value pair appearing in a BPEL element. For structured activities, the type of activity and its portion (i.e., *open* or *close*) appear in the third and fourth entries. Table II provides a portion of the BPEL artifact in its upper half, while its lower half shows its frame representation. This example pertains to the top two stack entries in Figure 1. This table also shows supporting declarations and assumptions. It will serve as an example of how our prototype performs translations into Promela.

### III. Frames to Promela

The lack of a direct mapping from BPEL to Promela initially motivated our use of some intermediate form  $R$ . Although we had not yet dealt with specifics, we extended  $R$  with modeling assumptions to obtain  $R'$ . From  $R'$  we produce derivative forms  $R_d$ ,  $R_s$ , and  $R_o$ , each suitable for generating Promela declarations, services, and the orchestration, respectively. We now discuss each encoded assumption within the context of this translation.

We generate a three-part Promela model of how BPEL orchestrates its interaction with its environment. The first part,  $R_d$ , includes declarations of message types, channels, and variables, each drawn from corresponding BPEL partner link and control link declarations. The second part,  $R_s$ , models each service that interacts with the orchestration, while the third part,  $R_o$  models the orchestration being verified. Unlike declarations, generating the latter two parts is more complex, since each Promela expression draws from two or three separate contexts.

Relation  $R_s \subseteq R'$  can be computed by selecting all

#### A BPEL code snippet:

```

17 <partnerLink name="pWhs" partnerLinkType="pltWhs"
24 <variable name="vPO" messageType="mtPO" />
26 <variable name="vShI" messageType="mtShI" />
27 <variable name="vAvl" messageType="mtAvl" />
41 <link name="xStI" />
43 <sequence>
44 <assign>
45 <copy>
46 <from>vPO.ci</from>
47 <to>vShr.ci</to>
48 </copy>
49 </assign>
50 <invoke partnerLink="pWhs"
51 operation="checkInventory"
52 inputVariable="vPO"
53 outputVariable="vAvl">
54 <sources>
55 <source linkName="xStI" />
56 </sources>
57 </invoke>
58 </sequence>
59 </sequence>

```

#### The corresponding relation $R'$ :

```

01 require basic atomic
01:17 partner plName pWhs
01:17 partner plType pltWhs
01:17 assume bufsize 1
01:24 variable vName vPO
01:24 variable msgType mtPO
01:26 variable vName vShI
01:26 variable msgType mtShI
01:27 variable vName vAvl
01:27 variable msgType mtAvl
01:30:36:41 link lDecl xStI
01:30:36:43 struct strucType sequenceOpen
01:30:36:43:44 basic actType assign
01:30:36:43:44 basic operation copy
01:30:36:43:44 basic from $vPO.ci
01:30:36:43:44 basic to $vShr.ci
01:30:36:43:50 basic actType invoke
01:30:36:43:50 basic pLink pWhs
01:30:36:43:50 basic portType pWhsPT
01:30:36:43:50 basic operation checkInventory
01:30:36:43:50 basic inVar vPO
01:30:36:43:50 basic outVar vAvl
01:30:36:43:50 basic source xStI
01:30:36:43 struct strucType sequenceClose
01:30:36:59 struct strucType sequenceOpen

```

TABLE II. BPEL (top) and its relation  $R'$  (bottom)

frames  $\mathcal{F}_p \in R'$  which for any frame  $\mathcal{F}_p$ , there exists some attribute  $a_p \in A$  that denotes partner link name. This, for example, includes attributes `plName` and `pLink` that appear in declarations or basic activities. Ordering these frames by their reference to partner link name, will group their frames together by service, resulting in  $R_s$ . Hence, generating Promela code for each service entails an in-order traversal of  $R_s$ . Generating the orchestration is somewhat simpler, since it entails traversal of  $R_o$  in its original lexical ordering.

## A. Formulating rules

Generating Promela code entails applying a number of rules to  $R'$ . Here we describe how rules are formulated.

Each rule is comprised of an antecedent and a consequent. The antecedent is a Boolean expression over a set of terms, where each term is an *assertion* involving one or more *slot references*. We treat each slot reference as an atom in a propositional logic, expressing it as some pair  $(t_i, a_i)$  where  $t_i \in T$  and  $a_i \in A$ . A primitive assertion may test for the *existence* of some tuple  $r_k = (c_k, t_k, a_k, v_k) \in \mathcal{F}_k$  which evaluates to true only if  $(t_i \equiv t_k) \wedge (a_i \equiv a_k)$ . A non-primitive assertion may involve some comparison between the *values* of two slot references. In addition to asserting the existence of each tuple in the comparison, assertion " $(t_i, a_i) \equiv (t_j, a_j)$ " further implies the equivalence of their *values*  $v_i \equiv v_j$  in tuples  $r_i = (c_i, t_i, a_i, v_i) \in \mathcal{F}_i$  and  $r_j = (c_j, t_j, a_j, v_j) \in \mathcal{F}_j$  respectively. If the Boolean expression over these assertions evaluates to 'true', then the antecedent is said to be *satisfied*, and all slots in all matching frames will become *visible* to the consequent.

The consequent is comprised of a sequence of slot references and unquoted string constants that specify the Promela expression to be generated. For slot references inside the consequent, our prototype outputs the value  $v_k \in V$  corresponding to some slot reference  $(t_k, a_k)$ . The following paragraphs describe a set of rules that generate Promela code from the lower half of Table II. Beneath each rule, we provide an example of its application.

## B. Generating declarations

Promela declarations collectively refer to message types, channels and variables. Their rules have antecedents comprised of a single primitive assertion. Rule 1 exhibits the most interesting consequent and is the one that generates channel declarations.

A closer look at channel declarations provides insight into assumptions concerning synchrony. Implied in the BPEL execution model is that each orchestrated web service operates asynchronously. We encoded this assumption as the slot 01:17 `assume bufsize 1` which matches slot reference `(assume, bufsize)` in Rule 1. Related work [2], [6] identifies the conditions under which one may model web services *as if* they operated synchronously. If one or more services in a composition is synchronizable, and if that portion was observed to be reliable, then we can realistically model it as a composition of synchronous *rendezvous* channels of buffer size zero. During verification, a model with such zero-place channels will tend to have a smaller and more tractable state space.

The model in our example was generated under optimistic assumptions. Pessimistic assumptions require introducing some notion of non-determinism to simulate

**Rule 1:**  $\exists (partner, plType) \Rightarrow$   
`chan (partner, plType) =`  
`[(assume, bufsize)] of {mtype, byte};`  
`chan plTWhs = [1] of {mtype, byte};`

**Rule 2:**  $( (partner, plName) \equiv (basic, plLink) ) \Rightarrow$   
`active proctype (partner plName)() { do :: {`  
`active proctype pWhs() { do :: {`

**Rule 2':**  $(partner, plName) \not\equiv (basic, plLink) \Rightarrow$  } od }  
`} od }`

fault-prone operation. This is needed to model cancellation of activities in BPEL, be they fault and compensation handlers or the `<terminate>` construct. Since cancellation of groups of activities can occur only within some scope, context must be propagated from BPEL to the machine verifiable model. At minimum, we must assume as did [9] that a cancellation requires a two-place buffer for each activity inside some scope. Additionally, cancellation requires generating Promela code for BPEL's default `catchAll` fault handler. Hence, when listening over its 2-place channel, a receiving activity must give cancellation-type messages priority. Cancellation is not the only use case for non-determinism. The BPEL `<pick>` activity requires that all non-selected activities affirmatively receive a skip message lest they hang [11]. Generating the Promela code under various sets of pessimistic assumptions is left for future work.

## C. Generating services

A Promela artifact must model how BPEL orchestrates its interaction with its environment. In the version of the Purchase Order Process used in our case study, the environment includes three services: an order processing service operating via partner link *pOP*, warehouse service via *pWhs*, and payment service via *pPay*. Each of these three services are modeled as a separate process (i.e., Promela *proctype*). A Promela process definition modeling a web service includes a process declaration, followed by a body that includes pairs of sending (!) and receiving (?) channel operators. Rule 2 generates the process declaration at the first frame for which the antecedent is satisfied. Its closing form, Rule 2' completes the process definition with a negated antecedent.

Basic activities like the ones in this case study must execute atomically [11], [16], requiring us to stipulate tuple 01 `require basic atomic`. It's context 01 makes it applicable to all activities in the composition. Guided by this requirement, Rule 3 and its closing form produce the Promela code to model this behavior. As pointed out by [3], [9], this assumption is not always tenable. Implementations of middleware layers can permit interleaved execution of more than one basic activity via the same partner link. Relaxing the atomicity assumption produces

**Rule 3:**  $\exists ( (require, basic, atomic) \wedge (partner, plName) \equiv (basic, pLink) \wedge (link, lDecl) \equiv (basic, target) \vee (variable, vName) \equiv (basic, inVar) \vee (variable, vName) \equiv (basic, outVar) \vee (link, lDecl) \equiv (basic, source) ) ) \Rightarrow$   
 atomic {  
 atomic {

**Rule 4:**  $( (partner, plName) \equiv (basic, pLink) \wedge (variable, vName) \equiv (basic, inVar) ) \Rightarrow (partner, plType)?(variable, msgType)((variable, vName));$   
 pltWhs?mtPO(vPO);

a more realistic model, but at the expense of a larger state space. If only one service has been observed to violate the atomicity assumption, it would be useful to model only its channel events as not being atomic. Thus we can include tuples of the form  $(c_j \text{ require basic atomic})$  for context  $c_j$  of each basic activity  $j$  for which the atomicity assumption holds.

Rule 4 generates an expression that listens over a channel (i.e., `pltWhs`) for a message (i.e., `mtPO(vPO)`) inside service `pWhs`. The sending end of this channel resides in the orchestration. It is generated by a complementing form that is otherwise identical to the listening end, except that the channel operator in the consequent is reversed. Notice that *input* and *output* variables are so named from the standpoint of the service rather than orchestration. Thus, it is the service that listens using the `?` operator for input variable `inVar`.

The remaining two channel expressions within the atomic scope of the service that supports the basic activity shown in Table III were each generated by its own rule. The rule for the first expression generates a send of an availability message (i.e., `mtAvl(vAvl)`) over a channel (i.e., `pltWhs`) back to the orchestration. The rule for the second expression generates code for the sending or *source* end of a control link (i.e., `ltxStI`). The control link is always synchronous and the expression for its source end is always placed last in any sequence of messages for a basic activity. Doing so blocks the start of the activity at the destination or *target* end until the activity at the source end completes. Since all three message events occur inside the same BPEL basic activity, and since we assume atomic execution, they are all placed inside the scope of the same atomic construct, as can be seen in the service portion of Table III.

#### D. Generating the orchestration

The orchestration is then modeled as a single process that mediates interaction between these services. Table III shows the Promela code for both the warehouse service and the portion of the orchestration that interacts with it.

Our prototype also automatically inserts Promela labels that support verification-time detection of deadlock and

---

```

Service:
active proctype pWhs ()
{
  do
  {
    atomic
    {
      pltWhs?mtPO(vPO);
      pltWhs!mtAvl(vAvl);
      ltxStI!xStI;
      pltWhs!mtShI(vShI);
    }
  }
  od
}

Orchestration (portion):

/* sequence43 start */
/* BPEL assign activity abstracted away */
atomic
{
  pltWhs!mtPO(vPO);
  pltWhs?mtAvl(vAvl);
}
/* sequence43 end */
...

```

---

TABLE III. Sample Promela output

progress. Declaration and maintenance of variables and their use in formulating orchestration-specific property assertions still require manual insertion, as is the formulation of properties in temporal logic. Further automation is left for future work.

#### IV. Related Work

The authors assessed tool support for the formal verification of safety or fiscally critical service oriented architectures (SOA) [14], e-science SOA [15], and use cases that demand automating the process of conversion [13]. To date, the most mature conversion tool support is offered by the WSAT utility [5] for converting BPEL to modeling languages for the Spin and SMV model checkers. However, the models generated do not make clear the assumptions used, nor do they appear to be intended for inspection. Thus, for example, an error trace from Spin becomes difficult for a human to interpret.

A prototype tool for translating BPEL into Promela was mentioned in [9]. It supports parameterization by degree of asynchrony based on a hierarchy of communication models. It generalizes on earlier work that identifies under what circumstances can a BPEL composition be treated as if it were a collection of synchronously communicating web services [6]. Of interest in [9], is their description of tool support for identifying the simplest model (i.e., in terms of queueing assumptions) that nonetheless retains some specified property (i.e., boundedness). This work did not address the atomicity assumption, nor was there a clear description of how it might be extended to address it.

A means of translating BPEL to Promela via an open workflow net was described in [10]. As part of the Tools4BPEL initiative <sup>2</sup>, BPEL2oWFN employs a form of flexible model generation into its intermediate form,

<sup>2</sup><http://www2.informatik.hu-berlin.de/top/tools4bpe/>

providing an approach to generate a compact model that is tailored to the analysis goal. The specifications were coarser-grained than our approach. Furthermore, the tool used for translating an open workflow net to Promela resulted in a single Promela *proctype*, which made it difficult to simulate.

Tool support for conversion from BPEL to a process algebraic formalism is offered by LTSA. LTSA uses an Eclipse plug-in to do this conversion into their formalism that can then be compared to that generated from user-specified message sequence charts [4]. It can model assumptions concerning the presence or absence of synchrony, atomicity, and determinism but it is not clear whether their tool supports scoping these assumptions to specific basic or structured activities.

Conversion of BPEL to workflow type Petri nets can model a robust set of concerns that involve all three classes of assumptions using their BPEL2PNML tool [11], [12]. This approach did not seek to exploit the explicit channel semantics of Promela, nor was it obvious if different parts of a composition can be subject to different assumptions.

Once in Promela, there are a number of tools for generating test suites [17] or for converting Promela to models suitable for other verification environments. One such tool offered by the VeriTech Project, provides a wide range of tool choices to manage, reframe, or effectively sidestep issues like the state space explosion problem [7]. In addition to the widespread use of Spin [8], the VeriTech tool motivated our choice of Promela by providing a gateway to other verification formalisms.

## V. Summary and Future work

We described an extendible mapping of BPEL artifacts to machine-verifiable models written in Promela – the modeling language used by the Spin model checker. Extendibility requires retaining the context of each attribute-value pair appearing in a BPEL artifact. Using a finite state transducer we described this notion of context, and by construction, defined the translation from BPEL into a well-defined intermediate form. This intermediate form supports inclusion of entries pertaining to assumptions that can be scoped to a portion of the composition. We then defined rules that generate Promela code from entries in the intermediate form. We used an excerpt from a well-known case study to illustrate these ideas.

Of the fifteen rules we already specified, we presented four of them along with either their closing or complementing forms. Specifying a rule set that is in some way *complete* is left for future work. This also entails extending our prototype accordingly and applying it to a suite of BPEL artifacts. Translations involving pessimistic assumptions will require modeling non-determinism.

## References

- [1] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guzar, N. Kartha, C. K. Liu, R. Khalaf, D. Knig, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web Services Business Process Execution Language Version 2.0, April 2007. <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>.
- [2] T. Bultan, J. Su, and X. Fu. Analyzing conversations of web services. *IEEE Internet Computing*, 10(1):18–25, Jan-Feb 2006.
- [3] H. Foster, W. Emmerich, J. Kramer, J. Magee, D. S. Rosenblum, and S. Uchitel. Model checking service compositions under resource constraints. In I. Crnkovic and A. Bertolino, editors, *In Proceedings of the 6th ESEC/SIGSOFT Symposium on Foundations of Software Engineering*, pages 225–234. ACM, 2007.
- [4] H. Foster, S. Uchitel, J. Magee, and J. Kramer. LTSA-WS: a tool for model-based verification of web service compositions and choreography. In L. J. Osterweil, H. D. Rombach, and M. L. Soffa, editors, *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, pages 771–774. ACM, 2006.
- [5] X. Fu, T. Bultan, and J. Su. WSAT: A tool for formal analysis of web services. In R. Alur and D. Peled, editors, *Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 510–514. Springer, 2004.
- [6] X. Fu, T. Bultan, and J. Su. Synchronizability of conversations among web services. *IEEE Transactions on Software Engineering*, 31(12):1042–1055, Dec. 2005.
- [7] O. Grumberg and S. Katz. VeriTech: a framework for translating among model description notations. *International Journal on Software Tools for Technology Transfer*, 9(2):119–132, 2007.
- [8] G. J. Holzmann. *The Spin Model Checker, Primer and Reference Manual*. Addison-Wesley, Reading, Massachusetts, U.S.A., 2003.
- [9] R. Kazhamiakin, M. Pistore, and L. Santuari. Analysis of communication models in web service compositions. In *Proceedings of the 15th international conference on World Wide Web (WWW'06)*, pages 267–276, New York, NY, USA, 2006. ACM.
- [10] N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg. Analyzing interacting WS-BPEL processes using flexible model generation. *Data & Knowledge Engineering*, 64(1):38–54, 2008.
- [11] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. *Science of Computer Programming*, 67(2-3):162–198, 2007.
- [12] B.-H. Schlingloff, A. Martens, and K. Schmidt. Modeling and model checking web services. *Electronic Notes Theoretical Computer Science*, 126:3–26, 2005.
- [13] J. C. Sloan and T. M. Khoshgoftaar. Toward model checking web services over the web. In *Proceedings of the Twentieth International Software Engineering and Knowledge Engineering Conference, (SEKE;08), San Francisco, California*, pages 519–524. Knowledge Systems Institute Graduate School, July 1-3 2008.
- [14] J. C. Sloan and T. M. Khoshgoftaar. Tradeoffs in testing service oriented architectures. In *Proceedings of the 14th ISSAT International Reliability and Quality in Design Conference, Orlando, Florida*, pages 141–145. ISSAT, August 7-9 2008.
- [15] J. C. Sloan, T. M. Khoshgoftaar, and V. Raghav. Assuring timeliness in an e-science service-oriented architecture. *Computer*, 41(8):56–62, August 2008. IEEE Computer Society.
- [16] W. M. P. van der Aalst, J. B. Jørgensen, and K. B. Lassen. Let's go all the way: From requirements via colored workflow nets to a BPEL implementation of a new bank system. In R. Meersman, Z. Tari, M.-S. Hacid, J. Mylopoulos, B. Pernici, Ö. Babaoglu, H.-A. Jacobsen, J. P. Loyall, M. Kifer, and S. Spaccapietra, editors, *Proceedings of the OTM Conferences (1)*, volume 3760 of *Lecture Notes in Computer Science*, pages 22–39. Springer, October 31 - November 4 2005.
- [17] Y. Zheng, J. Zhou, and P. Krause. A model checking based test case generation framework for web services. In *Information Technology: New Generations*, pages 715–722. IEEE Computer Society, 2007.

# Generating Test Cases of Composite Services Based on OWL-S and EH-CPN

Bixin Li<sup>1,2</sup>, Ju Cai<sup>1</sup>, Dong Qiu<sup>1</sup>, Shunhui Ji<sup>1</sup>, and Yuting Jiang<sup>1</sup>

<sup>1</sup>School of Computer Science and Engineering, Southeast University  
Nanjing 210096, Jiangsu Province, P.R.China Email: bx.li@seu.edu.cn

<sup>2</sup>Dept. of Computer Science and Engineering, University of California  
Riverside, CA92521, USA. Email: lbxin@cs.ucr.edu

## Abstract

*In web service times, the techniques for composing services are the base of service reuse and automatic integration. A new web service will be generated by composing some existed web services, these web services cooperate each other to provide a new more complex function. It is needed and very important to test the interaction behavior between any two web services during composition. In this paper, a kind of enhanced hierarchical color petri-net (or EH-CPN) is introduced to generate test cases for testing the interaction, where EH-CPN is transformed from OWL-S document, and both control flow and data flow information in EH-CPN are analyzed and used to generate an executable test sequence, and further test cases are created by combining the test sequence and test data in an XML file.*

## 1 Introduction

Web service technology has got widely and warmly welcomed in developing application software based on internet environment, but it has raised many new challenges for its testers, where two core challenges will be considered in this paper are: (1) source code of a web service is invisible to tester: clients of a web service can have functions provided by the service but they cannot get the source code of the service. It means that structure testing strategies are not able to be used to test your wanted single services, because it is impossible for testers who are not service providers themselves to generate test cases from source code of a web service, some informal specifications are explored to see the possibility for generating test cases, so both difficulty and complexity raise. The specifications which are being explored include WSDL, BPEL, and OWL-S, some necessary transforms are needed to generate test cases automatically and precisely; (2) many intermediate states of web service are also invisible to testers, it is hard to do testing manually, some automatic test techniques are needed.

In order to solve those challenge problems, researchers have introduced a variety of useful methods [1, 2, 3]. But most of these methods are based on WSDL or BPEL specification that specifies the location of the service and the operations (or methods), which the service will be exposed to clients. Test cases generated by these methods are more suitable for atomic web service or a composite service within an organization. In a wide composing service environment, web service is usually used to interact with other services from different organizations, it will play a different role. It is needed and very important to test the interaction behaviors among different web services when we compose some existed web services in a certain style to provide new functions.

In this paper, an EH-CPN based test case generation approach has been introduced, where EH-CPN is an enhanced hierarchical color Petri Net. The outline of the approach is summarized as follows: at first, OWL-S document is transformed to EH-CPN and further the process of web service composition is displayed by this kind of Petri Net; next, data flow and control flow information of EH-CPN are analyzed in detail to find all *output-input-define-use chains* (or *OI-du-chain*); next, *OI-du-chain* is extended to correspondent executable test sequences satisfying ALL-DU-PATHS criterion; finally, both test sequences and test data are combined to generate test cases.

## 2 Primaries

There are three important concepts will be used in this paper, let's see how they are defined.

**Definition 1** *Multiset*<sup>[4]</sup>

A multiset *bag* is a function from a non-empty set  $A$  to non-negative integer set  $IN$ ,  $bag : A \rightarrow IN$ . Let set  $Bag(A) = \bigcup_{a \in A} bag(a)$  be the set of all the multisets that are defined in set  $A$ .

**Definition 2** *E-CPN*

An extended color Petri Net (marked as E-CPN in this paper) is defined as follows: E-CPN is a 6-tuple  $\langle P, T, C, Cd, Pre, Post \rangle$

(1)  $P$  is a finite set of places;

(2)  $T$  is a finite set of transitions. There are five kinds of typical transitions in web service composition: (2.1) *service invoking transition*: When this transition is fired, it will invoke the corresponding web service; (2.2) *condition controlling transition*: When this transition is fired, it will invoke a condition checking function whose return value is Boolean, and which places the transition will go to depends on the return value; (2.3) *concurrent controlling transition*: This transition is used to assort with synchronization between transitions; (2.4) *interface transition*: This transition will invoke checking function to check whether or not the output from upper net equals to the input to fire sub net; (2.5) *empty transition*: This transition will invoke nothing. The aim to define this transition is to make the net satisfy the definition of Petri Net in some special condition.

(3)  $C$  is a finite set of colors.

(4)  $Cd$  is a color function  $Cd : P \cup T \rightarrow C$

(5)  $Pre, Post \in \beta^{|P| \times |T|}$ , both are Incidence Matrixes, where  $Pre$  is a pre-Incidence Matrix and  $Post$  is a post-Incidence Matrix satisfying following equations:

$$\forall (p, t) \in P \times T, Pre[p, t] : Cd(t) \rightarrow Bag(Cd(p)),$$

$$Post[p, t] : Cd(t) \rightarrow Bag(Cd(p));$$

$\beta$  is a set of grouping functions in following form:

$$\beta : Cd(t) \rightarrow Bag(Cd(p))$$

### Definition 3 EH-CPN

An enhanced hierarchical color Petri Net (marked as EH-CPN in this paper) is defined as follows: EH-CPN is a 4-tuple  $\langle S, C, IC, I_0 \rangle$

(1)  $S$  is a finite set of sub nets satisfying following features:

(1.1)  $\forall s \in S, s = (E-CPN, Ci, Co)$ ,  $Ci$  is a finite set of input colors,  $Co$  is a finite set of output colors.

(1.2)  $\forall s_i, s_j \in S$  and  $s_i \neq s_j, (P_{s_i} \cup T_{s_i} \cup A_{s_i}) \cap (P_{s_j} \cup T_{s_j} \cup A_{s_j}) = \Phi$

(2)  $C$  is a finite set of color

(3)  $IC$  is an interface checking function which will check whether or not the output coming from the upper net equals to the input that can fire the sub net.

(4)  $I_0$  is an initialization state

## 3 Transforming OWL-S to EH-CPN

OWL-S is a web service describing language based on ontology. OWL-S document contains some useful control

flow and data flow information, but they all are hidden in the descriptive level document. In order to analyze and capture some useful information, it is needed to introduce a mechanism so as to transform OWL-S document to an EH-CPN in constructive way. In this mechanism: (1) all input variables and output variables are represented by *color tokens*, where each variable has its own *color*; (2) the services are transformed into *transitions*; (3) all input and output states are transformed into *places* containing *tokens*; (4) the pre-condition is represented by a *condition checking function* in condition controlling transition or a *guard function*; (5) the effect is represented by an *output arc*.

In OWL-S document, some outputs are conditional output, which means the output will contain different variables according to different conditions. To transform these outputs into corresponding places, we use a *condition controlling transition* to follow the place to control different outputs. We also change the condition expression to condition checking function and put the function into the condition controlling transition. In this way, different outputs will be dispatched to different places. The control flow and data flow relations will be captured by *connecting arcs*, *transitions* or *arc expressions*.

In OWL-S specification, the service process is divided into three kind of forms, where the atomic process and the composite process can be invoked, but the simple process can not be, and therefore the transformation of simple process isn't needed. So we only analyze atomic process and composite process in next sections.

### 3.1 Atomic process transformation

The atomic process describes the process of single service that means it can not be divided again. It also has no sub-services. When the input satisfies the firing rule, the process will be invoked and corresponding output will be produced. The construction of the atomic process of sub net is more complex than that of the upper net, because sub net must be connected to its supper net based on some conditions. In order to check whether the input tokens coming from upper net is conformance to the tokens required by sub net, we add an *interface transition*. In this transition there is an *interface checking function* used to check the input. But we do not need this transition in non-sub nets.

In this paper, the atomic processes are divided into four types according to the input and output: (1) *Input coming from single net*. In this case, the input only comes from upper net. The transformation refers to Figure 1(a). (2) *Input coming from multi-nets*. In this case, the input comes from the upper net and the local net. The transformation refers to Figure 1(b). (3) *Non-conditional output*. The transformation of the output part is the same as the output part of Figure 1(a). (4) *Conditional output*. We add condition con-



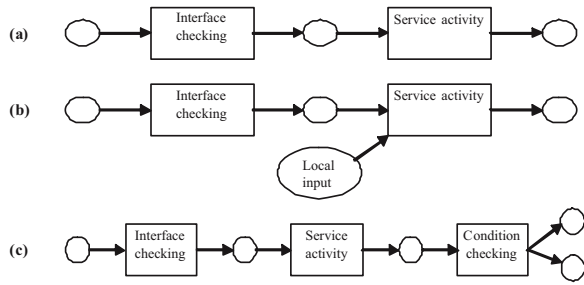


Figure 1. The construction of atomic process



Figure 2. Sequence structure

trolling transition to dispatch different output according to the result of condition checking. The transformation refers to Figure 1(c).

### 3.2 Composite process transformation

A composite process can be decomposed into some atomic processes and/or other smaller composite processes. If we organize atomic processes or composite processes in a certain order using some control constructs, we will have new web services for providing new functions. The control constructs used in composite process include: sequence, split, split+join, choice, any-order, if-then-else, iterate, repeat-while, and repeat-until etc. Now we discuss how to transform composite processes to EH-CPN in detail.

**Sequence:** The processes contained in this construct will be invoked sequentially. The transformation refers to Figure 2

**Split+join:** In this construct, the concurrent processes are described too. All the processes have not only the same precursor but also the same subsequence. When all the concurrent processes end, they enter next state at the same time. We use a *concurrent controlling transition* to coordinate this kind of synchronization. The transformation refers to Figure 3

**If-then-else:** Three properties i.e., *ifCondition*, *then* and *else*, and two kind of services components are contained

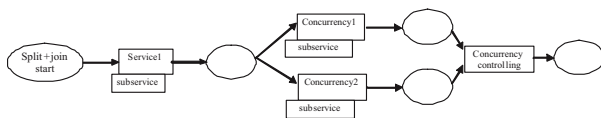


Figure 3. Split+Join structure

in this construct; If *ifCondition* is true, the service in *then* branch will be executed; otherwise, the service in *else* branch will be executed. In our transformation, we map the property *ifCondition* to a *condition controlling transition* in EH-CPN to dispatch different states.

**Repeat-while:** In this construct, one *testing condition* and one *loop-process* are contained. It tests the condition, then, does the loop-process if the result is true, exits else. So the loop-process is not executed if the condition is false. In our transformation, the testing condition is mapped to *condition checking function*, and then the function is put into a *condition controlling transition* to display this construct in EH-CPN.

**Repeat-Until:** This construct contains one testing condition and one *loop-process*, which is the same as Repeat-While construct. But there is a little difference of the execution process between them, it executes the loop-process first, then checks the condition, later the loop continues if the condition is true, exits else. So the loop-process will be executed at least once anyway. We also map the testing condition to a *condition checking function* and put it into *condition controlling transition* to display this construct in EH-CPN.

**Any-order:** This construct contains a list of processes which will be invoked in any order except for concurrency. All the processes must be executed at least once.

Using the above mechanism, we transform the control constructs in OWL-S document to EH-CPN in a constructive way. So the EH-CPN can intuitively describe the *control flow* of one process. But *data flow* is not very obvious.

## 4 Data flow in EH-CPN

In this section, we will discuss how to capture data flow and control flow in the EH-CPN.

### 4.1 Notations and definitions

Firstly, we need to clarify some useful notations and definitions that will be used in our analysis.

(1) During the transferring process from transition  $T_i$  to place  $P_j$ , the transition  $T_i$  will send some tokens  $(x_1, x_2, \dots, x_n)$  to its subsequence place  $P_j$ . In EH-CPN, these tokens are the interactive output of transition  $T_i$  and are defined in place  $P_j$ . We mark this relation as  $T_i \cdot P_j(x_1, x_2, \dots, x_n)$  in EH-CPN.

(2) During the transferring process from place  $P_i$  to transition  $T_j$  (non condition-controlling transition), the transition  $T_j$  will receive all needed tokens, which form the interactive input of transition  $T_j$ . Obviously, they are computation-use (or c-use) in  $T_j$ , and this relation is marked as  $P_i \cdot T_j(x_1, x_2, \dots, x_n)$  in EH-CPN.

(3) If transition  $T_j$  is a condition-controlling transition, it will judge all the tokens  $(x_1, x_2, \dots, x_n)$  from pre-place  $P_i$ . Obviously, these tokens are predicate-use (p-use) in  $T_j$  and this relation is also marked as  $P_i \cdot T_j(x_1, x_2, \dots, x_n)$  in EH-CPN.

(4) If a token (i.e., a variable)  $x$  is defined in place  $P_i$  and used in transition  $T_j$  (c-use or p-use), we call  $(P_i, T_j)$  a *define-use pair* of token  $x$  and mark it as  $(P_i, T_j)_x$  in EH-CPN.

(5) A *path segment* in EH-CPN is defined as a sequence which is composed of places and transitions and marked as  $PATH = (P_i, T_i, \dots, T_j, P_j, \dots)$ , where,  $P_i, P_j \in P$ ,  $T_i, T_j \in T$ . If token  $v$  is defined in place  $P_i$  and used in transition  $T_j$ , we define  $PATH(v)$  as a *define-use path segment* for  $v$ . If token  $v$  is defined only in a place  $P_i$  of  $PATH(v)$  and no redefinition in any other places, we define this path as *def-clear-path segment* for  $v$ .

(6) From first three items, we know that all tokens are the output of their pre-transition, and meanwhile the input of their subsequence transitions. If a token  $v$  is defined in place  $P$  as the output of the pre-transition of  $P$  (marked as  $O$ ) and is used in transition  $T$  as the input of the transition  $T$  (marked as  $I$ ). Place  $P$  and transition  $T$  are in the same path (marked as  $PATH_{PT}$ ). We define this path as the output-input-define-use chain (*OI-du-chain*) for token  $v$  and mark this relation as  $(O, PATH_{PT}, I)$ .

If a token is used in a transition, it will be consumed, so a token can be used only once in one process. Different services will produce different outputs, so every token will be defined only once according to first item. Therefore, we can conclude that every *OI-du-chain* is a *define-clear-path segment*.

(7) For a given set of test data, if there is a path in EH-CPN, each transition in this path will be triggered sequentially according to the order in path and arrive at final designated position. We regard this path as an *executable path*.

(8) There are two kinds of special positions in Petri Net. One is the position that has no output-edges; another is the position with *end* label. Both of them are regarded as *end position* in EH-CPN.

## 4.2 Data flow analysis

By analyzing the incidence matrixes of EH-CPN, we will have some useful data flow information for test case generation.

(1) we will have the *define-use pairs* of all tokens. On one hand, the post-incidence matrix records tokens which are produced after transitions have been fired. So a token  $v$  is defined at the place where token appears for the first time and this place should be added in the *define-use pair* of token  $v$ . On the other hand, the pre-incidence matrix records tokens which are needed to fire transitions. So the

transition which requires token  $v$  for the first time and this transition should be added in the *define-use pair* of token  $v$ .

(2) After that, we can use those *define-use pairs* to find all *define-use-paths*, further we will have all *OI-du-chains* which contain all kinds of data flow information. In the *OI-du-chain*, we can see which services are affected by a certain variable. If we find all the *OI-du-chain*, we can get all the interaction influence between services.

In next section, we will discuss how to generate test case and illustrate how to get *OI-du-chain* in detail.

## 5 Test case generation

In EH-CPN based approach, test cases are generated in three phases: we will discuss how to produce test sequence in phase 1, then we discuss how to prepare test data in phase 2, finally we discuss how to generate test cases by combining test sequences and test data.

**Phase 1: generation of test sequence** Test sequences is generated according to following steps:

*Step 1: Preprocessor of EH-CPN* The main work is to identify the concurrent modules and modify them. In EH-CPN, the concurrent module is described in *split+join* structure. In this structure, there is a transition whose in-degree is one and out-degree is bigger than one. We call this transition *split transition*. There is a *synchronization controlling transition* whose out-degree is one and in-degree is bigger than one. Every concurrent module begins with a *split transition* and ends with a *synchronization controlling transition*. So we take this kind module as a black-box and use one transition to replace the module whose input is the input of *split transition* and whose output is the output of *synchronization controlling transition*. The algorithm for identifying all concurrent modules is omitted because the space limitation.

*Step 2: OI-du-chain generation* *OI-du-chain* is composed of three parts:  $O$ ,  $I$  and  $PATH$ . By analyzing incidence matrixes of EH-CPN, we can find all define-use pairs, and further we can determine the  $O$  and  $I$  for every token. The algorithm for computing the  $PATH$  of *OI-du-chain* consists of two phases: (1) the algorithm is used to find a sequence that begins with a transition where token  $v$  is used and ends with a place where the token  $v$  is defined; (2) the algorithm is used to reverse this sequence generated in (1). By this way, we get a sequence which is the path of the *OI-du-chain* for variable  $v$ .

*Step 3: Pre-sequence and post-sequence generation* To let the path of *OI-du-chain* be an executable path, we need to extend it with a *pre-sequence* and a *post-sequence*.

The computation of *pre-sequence* is easy, we can use the algorithm in step 2 to compute it as long as we use the start node of the EH-CPN and the first node in the path of *OI-du-chain* as the two input parameters respectively.

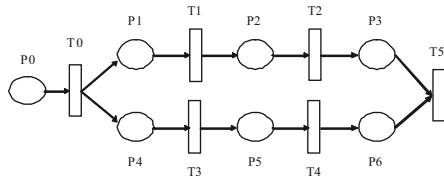


Figure 4. concurrent module

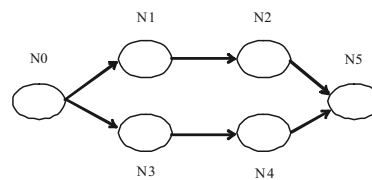


Figure 5. action graph

Post-sequence can be found as follows: firstly, we will find every subsequence node starting from the last node of the path of the *OI-du-chain* until we arrive at *end position*, and then, collect all the subsequence nodes orderly and a post-sequence will be found. If a node has more than one post-sequence, this way will find all the post-sequences.

Because EH-CPN is a hierarchical Petri-net, it is likely that a complete path, consisting of the path segment of one *OI-du-chain* and its *pre-sequence* and *post-sequence*, will contain some transitions to sub nets. In this case, it is necessary to replace those transitions with a new path in their sub nets using above steps 1-3. The algorithm will repeat this replacement process until the path segment of one *OI-du-chain* has been found, where transitions do not contain sub nets. In our EH-CPN based method, the path segment of *OI-du-chain* with its pre-sequence and post-sequence altogether are regarded as a *test sequence*.

*Step 4: Test sequence generation for concurrent module*  
After steps 1-3, we have got a test sequence, but this sequence is generated based on modified EH-CPN in step 1. where we just regarded concurrent module which includes many transitions and places as a transition simply for easy to deal with. If we find a sequence for real concurrent modules, we will have a complete and precise sequence based on the primary EH-CPN. In order to get test sequences from concurrent modules, we can do as follows: (1) we combine every transition and its pre-places into one node, maintaining relations between transitions unchanged. In this way, the concurrent module has only one kind of nodes and we name this net as *action graph*. (2) we construct test sequence tree. The tree contains all the concurrent test sequences. (3) one path which is from root to one leaf is a *test sequence*.

The following process illustrates how to transfer an *action graph* into a *test sequence tree*: (1) Make the node which is composed of *split transition* and its pre-place be the root of a *test sequence tree*; (2) Delete above nodes and their post arcs in *action graph*. The nodes which will be deleted are in a path from root to node  $i$  in the  $k^{th}$  ( $k \geq 0$ ) level; (3) Find nodes which have no direct precursor and let them be the children of node  $i$ .

Figure 4 shows the concurrent module in EH-CPN and Figure 5 is an action graph of Figure 4, where we can see that some transitions and places in Figure 4 have been

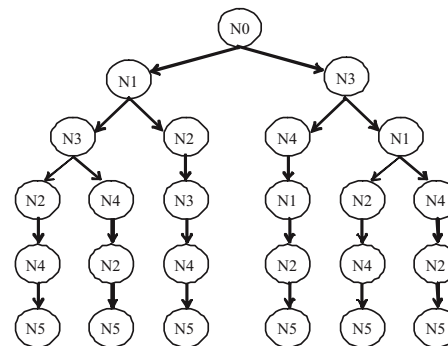


Figure 6. test sequence tree

united as a new node in Figure 5. For example,  $P1$  and  $T1$  are united as node  $N1$ .

Figure 6 is a test sequence tree coming from Figure 5.  $N0$  is composed of split transition and its pre-places. So it is the root of the test sequence tree. If we delete  $N0$  and its post arcs, we will find  $N1$  and  $N3$  with no precursor. So  $N1$  and  $N3$  are children of  $N0$  according to rule 2.

In Figure 6, the test sequence tree has six leaves, so it has six test sequences. If we map nodes in one test sequence of test sequence tree into corresponding places and transitions in EH-CPN, we will get the test sequence of concurrent modules. For example,  $(N0, N1, N3, N2, N4, N5)$  can be mapped to  $(P0, T0, P1, T1, P4, T3, P2, T2, P5, T4, P6, P3, T5)$ . After generating test sequence of concurrent module, we use these sequences to replace the corresponding modified transitions in step 1. In this way, we will get test sequences based on primary EH-CPN.

*Step 5: Executable test sequence generation* If there are loops in EH-CPN, the test sequence is likely non-executable. Because it is impossible to know how many times the loop will execute exactly. In our EH-CPN based way, we borrow the heuristic method, which is proposed by C. Bourhfir[7], to solve this problem by finding an appropriate loop and inserting it into the non-executable test sequence to generate an executable sequence. Now we get executable test sequences satisfying ALL-DU-PATH criterion.

**Phase 2: preparation for test data** The main idea to generate test data is originated from the XPT method

(XML-based Partition Testing) method in [8], which is consisting of three parts: (1)map XML Schema which defines the structures and data types of input and output of all the web services to *Category Partition*. In this way, a set of final instances and intermediate instance frames have been got; (2) find all the preconditions which are included in the services in one test sequence and do AND operation on all those preconditions to get the value domain of the instances; (3) generate the test data using random methods.

**Phase 3: generation of test case** Test cases in EH-CPN based way is the combination of test data and test sequence, where test sequence can be generated in section 5.1 and test data can be got using the way in section 5.2. In EH-CPN based way, the test sequence is only composed of all the web services contained in a test sequence generated in section 5.1. Test cases are coded in an XML file and can be used as an input of a test tool.

## 6 Conclusion

There are many methods have been proposed to generate test cases for web service. These methods can be parted into two basic categories: one can generate test cases based on specifications, the other can generate test case based on model checking, such as [1], [2],[4], [5], [6], [9], [10], and [11] etc. In this paper, we introduced a method to generate test cases based on a kind of extended hierarchical colored Petri Net, where we transform OWL-S to EH-CPN for capturing more control flow and data flow information so that we can generate more precise test case. But the problem is if there are too many services in one concurrent module, the state explosion problem rises, so it is necessary to find an effective method to solve state explosion problem and improve our method in future work.

## Acknowledgement

Bixin Li is now with University of California at Riverside as a visitor scholar and he thanks Prof. Rajiv Gupta in University of California Riverside for providing a very comfortable Lab. This work is partially supported by the National Nature Science Foundation of China under No.60773105, partially by the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2007513, and partially by National High Technology Research and Development Program under Grant No. 2008AA01Z113.

## References

[1] X. Y. Bai, W. L. Dong, W. T. Tsai, and Y. N. Chen. *WSDL-Based Automatic Test Case Generation for Web Services Testing*. Proceedings of the 2005

IEEE International Workshop on Service-Oriented System Engineering (SOSE'05).on 20-21 Oct. 2005 Page(s):207-212

- [2] Y. B. Wang, X. Y. Bai, J. Z. Li, and R. B. Huang. *Ontology-Based Test Case Generation for Testing Web Services*. Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07).on 21-23 March 2007 Page(s):43-50.
- [3] H. M. Sneed and S. H. Huang. *WSDLTest-A Tool for Testing Web Services*. Eighth IEEE International Symposium on Web Site Evolution, 2006. Sept. 2006. Page(s):14-21
- [4] Y. P. Yang, Q. P. Tan, Y. Xiao, J. S. Yu, and F. Liu. *Exploiting Hierarchical CP-Nets to Increase the Reliability of Web Services Workflow*. In: Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'06). on 23-27 Jan. 2006 Page(s):7 pp.
- [5] Y. P. Yang, Q. P. Tan, J. S. Yu, and F. Liu. *Transformation BPEL to CP-Nets for Verifying Web Services Composition*. Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05). On 22-26 Aug. 2005 Page(s):6 pp.
- [6] S. Y. Wang, P. Yu, J. J. Huo, and C. Y. Yuan. *Petri Nets for Systems Engineering*. Publishing House of Electronics Industry.2005.
- [7] C. Bourhfir, R. Dssouli, E. Aboulhamid, and N. Rico. *Automatic executable test case generation for extended finite state machine protocols*. In Proceedings of IWTCS'97 [17], pages 75-90.
- [8] B. Antonia, J.H. Gao, M. Eda, and P. Andrea. *Automatic Test Data Generation for XML Schema-based Partition Testing*. Automation of software Test 2007. Second International Workshop on 20-26 May 2007 page(s):4-4.
- [9] D. Martin, A. Ankoleka. *CongoProcess.owl document*. <http://www.daml.org/services/owl-s/1.2/>
- [10] Y. Y. Zheng J. Zhou P. Krause. *A Model Checking based Test Case Generation Framework for Web Services*. Fourth International Conference on Information Technology (ITNG'07). on 2-4 April 2007 Page(s):715 - 722.
- [11] L. Hua, and Y. X. Ming *Generation Executable Test Sequence Based on Petri-net for Combined Control and Data Flow of Communication Protocol*. International Conference on Communication Technology. On 22-24 October,1998 Page(s):S48-02-1 - S48-02-5

# User Perceived Response-time Optimization Method for Composite Web Services

Junfeng Zhao<sup>1,2</sup>, Yasha Wang<sup>1,2</sup>, Bing Xie<sup>1,2</sup>

<sup>1</sup>(Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China)

<sup>2</sup>(Software Institute, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

E-mail: { zhaojf, wangys, xiebing }@sei.pku.edu.cn

## Abstract

*A Composite web service's response-time is composed by the response-time of its member web services. In order to make a composite web service meet its user's response-time requirement, the instance of each member web service should be properly selected and bound. In the literature, every member web service is treated in the same way in the composite web service's response-time optimization and its different weight on the effect of user's satisfaction is not considered. However, in a certain scenario, some member services are more sensitive on response-time than others, that is to say, in a given composite web service, when these sensitive services delayed, the decline of users' satisfaction is remarkably greater than other services delayed. In this article, a user perceived response-time optimization method is proposed to reduce the delaying risk of the time sensitive web services, and thus to improve the user's satisfaction of a composite web service. Our experiments validated the efficiency of the proposed method.*

## 1. Introduction

As for service based application, response-time is one of the key QoS (Quality of Service) attributes that mostly affect user's satisfaction. A composite web service is a composition of some member web services, and each member web service often have more than one available instances which have the same functionality and different response-time. The response-time of a composite web service is composed by the response-time of its member web services. In order to make a composite web service meet to its user's response-time requirement, response-time optimization methods for selecting proper instance of each member web service should be applied.

In the literature, every member web service is treated in the same way in the composite web service's response-time optimization, but without considering their different weight on the effect of user's satisfaction. However, in a certain using scenario, some member services' response-time are more sensitive than others', that is to say, in a given composite web service, when these sensitive member services delayed, the decline of users' satisfaction is remarkably greater than when other member services delayed. The article will give an example in section 3 to show this.

In order to reduce the lateness risk of those sensitive member web services, and thus improve users' satisfaction of the composite web service, a user perceived response-time optimization method is proposed in this article. It identifies the sensitive web services in a given scenario, and assigns higher priorities on them in the on-time-assurance mechanism. During the runtime of a composite web service, the pre-planned selection of service instances for some member services may be invalid because of their response-time and accessibility changing, and thus a replanning for the un-executed member web services must be performed. Therefore a simple and fast dynamic replanning algorithm to meet with its real-time requirement is also given by the article.

The remainder of the paper is organized as follows: section 2 explains the preliminary of the method that will be given in the article. Section 3 introduces user perceived response-time optimization method for composite web services. Section 4 introduces an experiment and analyzes the result of experiment. The related works and their difference with the work in this article are discussed in section 5. Finally conclusions and future works are discussed in section 6.

## 2. Preliminary

The article specifies composite web services as a group of *abstract tasks* that linked with control flows and data flows. Abstract task is called *task* for short, it's an abstract of web service that has specific functions. The web service that realize some functions of a task are called a *service instance* of the task. Each service instance has same functions but different response-time. At runtime, response-time of a task is the response-time of the service instance that is bound with it.

Basically, we adopt the state chart like [1] to represent business process of composite web services. In order to make the conditional structure more clear, we add a lozenge symbol to connect different conditional branches. In our state charts, transitions represent data flows and control flows among tasks and can be labeled with events, conditions, and operations. States include atomic states and compound states. Atomic states are labeled with the identifier of the tasks they represent. When a atomic state is entered, the task it identifies is bound with a specific service and invoked by the service. Compound states include OR-states and AND-states. An OR-state contains a single region whereas an AND-state contains several regions (separated by dashed lines) which are intended to be executed concurrently.

Figure 1 demonstrates an example of the specification of an online bookstore composite web service whose process is simplified for discussion facilitation. At first, a customer can query books using this service. Then the result of the query is sorted according to his/her preferences. After received the query result, the customer can either add books into a shopping cart or launch another inquiry. After the last selected book is put into the shopping cart, a check-out process which includes two parallel threads will be started. In one thread, the distribution stores whose repertoires have the customer desired books are found and then a delivery plan, which includes the information of proper distribution stores, will be optimized according to the customer and the distribution stores' locations. In the other thread, the discount of each book is calculated according to the customer's credit level. Once both threads have been finished, a detailed order form will be send to the customer for confirming. The customer can choose to modify, cancel or confirm the order. Finally, a book-delivery-notice is sent to the distribution stores according to the delivery plan.

### 3. User perceived response-time optimization method for composite web services

The user perceived response-time optimization method includes three sequential steps which are time critical section identification, time critical section decomposition, and run-time service-instances selection. The two former steps are performed before composite web service's execution, while the last step is performed at run-time. In order to make the run-time algorithm simple and fast and thus meet with its real-time requirement, we put most complex calculations into the two former steps.

#### 3.1 Time Critical Section Identification

Generally, during the execution of a composite web service, users may interact with the service several times and wait for its feedback in each interaction. In a given interaction, the time from a user submits a request to the composite service till the request is fulfilled or the service feedback is sent back is called a *waiting cycle*. In a given *waiting cycle*, a fragment of the composite service's execution process is performed which includes one or more tasks. We call this fragment a *time critical section (TCS)*. One *TCS* is always relative to a *waiting cycle*. The length of a *TCS*'s related *waiting cycle* can be called the response time of this *TCS*. *TCS*s have two important attributes, *maximum-acceptable-time (MAT)* and *sensitive-level (SL)*.

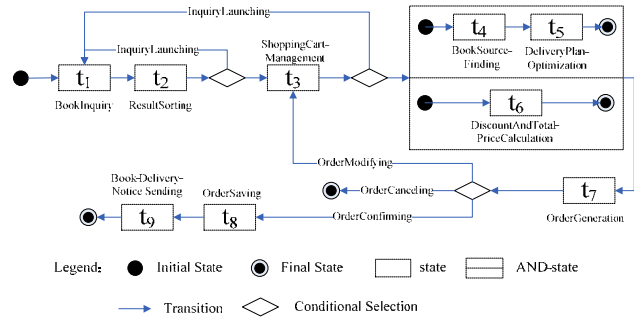
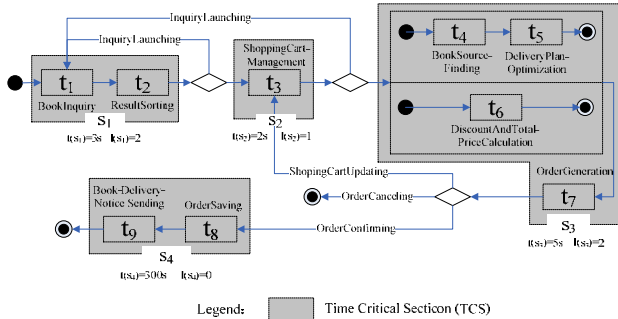


Figure 1. State chart of an online bookstore composite service

*MAT* describes users' requirement for response-time of a *TCS* in a specific scenario. As for *TCS*  $s$ , its *MAT* is denoted as  $t(s)$ . If the response-time of  $s$  exceeds  $t(s)$  in a given execution of the *TCS*, the users' satisfaction degree of the composite service would decline. *SL* is a non-negative integer to describe the sensitive degree of a *TCS*. The more sensitive a *TCS* is, the greater its user's satisfaction declines when its response time exceeds its *MAT*. Here we assign bigger *SL* to a *TCS* when it is more sensitive. *MAT* and *SL* can be set by domain experts according to domain knowledge of the composite service's domain.

*TCS* build a mechanism to combine users' perspectives of a composite service to providers' perspectives in response-time optimization. Usually, Service users are only aware of services' external characteristics of response-time, i.e. the length of *waiting cycles*, but don't care about their internal details, such as how many and what tasks contribute the *waiting cycle*. On the other hand, service providers know those internal details including the services' execution process, tasks, available service instances and their response-time, but generally don't know or don't use the information of users' feeling about services' response-time in response-time optimization. As improving users' satisfaction degree is the mainly purpose of response-time optimization and other QoS optimization activities, not only the internal details but also the information of users' feeling of a composite service are important and useful in response-time management. *TCS* contains information from both users' and providers' perspectives. *Waiting cycles* identified by *TCS*s, *MAT* and *SL*s of *TCS*s are information from users' perspective, while tasks contributing to *waiting cycles*, and the relationships among these tasks in the composite service's execution process are information from providers' perspective. The utilization of *TCS*s in response-time management helps increasing users' satisfaction. By utilizing identified *TCS*s and assigning their *MAT* service, provider can make individual *waiting cycles* meet with users' requirements by perform management activities in the scope each *TCS*'s tasks, thus not only the total response-time but also all *waiting cycles* of a composite service is managed. Furthermore, the *SL* of *TCS*s can help service providers prioritize *TCS*s according to users' feeling in response-time management, and thus reduce the lateness risks of the more sensitive *TCS*s.

Figure 2 demonstrate the *TCS* identification of the online bookstore showed in figure 1. Four *TCSs* are identified, and each one has a related *waiting cycle*. These four *waiting cycles* are the query-condition-input till query-result-gotten *waiting cycle*, books-selected till books-added-into-shopping-cart *waiting cycle*, check-out-started till order-list-gotten *waiting cycle*, and the order-confirmed till book-delivery-notice-received *waiting cycle*. In this example, there are three different *SLs* for *TCSs*, which are level 0, 1, or 2. According to domain knowledge, we set *MAT* of  $s_1, s_2, s_3$  and  $s_4$  as 2s, 2s, 3s and 300s, and set their *SLs* as 2, 1, 2 and 0 respectively.



**Figure 2. *TCS* identification of the online bookstore composite web service**

To set  $l(s_4)=0$  means  $s_4$  isn't sensitive to response-time. In online bookstore scenario, a book-delivery-notice are required to be sent to the distribution stores within 300s after the customer's confirming of a book order. But according to domain knowledge we know that the lateness of the notice's arrival won't cause much dissatisfaction of users. The reason may be the interaction between delivery-scheduling-men (i.e. the user who is response for checking the book-delivery-notices in distribution stores) and the composite web service is asynchronous. In other words, a delivery- scheduling-man won't predict when a notice will arrive, and sit beside a computer to wait for it. It need to point out that  $l(s_4)=0$  doesn't mean  $s_4$ 's response-time can exceed its *MAT* arbitrarily, but it means the excess won't cause serious troubles.

We set higher *SLs* to  $s_1$  and  $s_3$  than  $s_2$ . As  $s_1$  is similar to  $s_3$ , we only discuss the reason for setting the *SLs* of  $s_1$  and  $s_2$ . A customer can't start checking out until all selected books are added into the shopping cart successfully, so the customer is compelled to wait if he/ she checks out before  $s_2$  finishing. If he/she waits too long (the length of the wait cycle exceeds  $t_2$ 's *MAT*), his/her satisfaction degree will declined obviously. This shows  $s_2$  is more sensitive than  $s_4$ , and we should set  $s_2$  a higher sensitive level. In further analysis we find that not every time a customer chooses to check out after selecting books from a query result, instead, in many cases, he/she launches another inquiry for searching more books. In these situations, customers may be imperceptive of the lateness of  $s_2$  and the degree of satisfaction won't decline. In the contrast with  $s_2$ , the lateness of  $s_1$  always leads to users' obvious dissatisfaction, because users have no choice after submitting their inquiry request but waiting. In addition, in the example online book

store scenario, after users' delivering of their check-out request, the information of books in the shopping cart will be listed onto users' screen according to the data in a local buffer, users can check the book list while  $s_2$  is executing. As a contrast, users can't receive any feedback and have nothing to do when they are waiting for query results. So lateness of  $s_1$  leads to more complaint than  $s_2$ . The above discussion shows that  $s_1$  is more sensitive than  $s_2$ , so we set  $s_1$  a higher *SL* than  $s_2$ .

### 3.2 Time Critical Section Decomposition

A *TCS*'s response-time is composed by the response-time of its tasks'. The calculation of a *TCS*'s response-time by knowing its tasks' response-time is complicated because the execution process of the *TCS* may include complicated structures like cycle, parallel and conditional selection etc. The complexity of the calculation results in the complexity of run-time replanning algorithms which are applied when some service instances' response-time and accessibility change during the execution of a composite web service. In order to make the run-time algorithm in our method meet its real-time requirement, we decompose each *TCS* into several *sub-TCSs* beforehand.

The decomposed *sub-TCSs* should meet the following two conditions:

1. A *TCS* should be decomposed into a set of *sub-TCSs* which is a partition of the *TCS*, that is to say, every task in the *TCS* is contained and only contained by one *sub-TCS*;
2. The execution time of each *sub-TCS* is the sum of response-time of its tasks, that is to say, a *sub-TCS* is made up of a set of sequential executing tasks.

There are three steps to decompose a *TCS*.

Step one: cycle unfolding. We use the method stated in [2] to unfold cycles in a *TCS*. The method first calculates the average number of rounds that each cycle is executed in a composite service according to historical data of the service's execution, which we denote as  $n$ . Then in the following calculation, response-time of each task in the cycle is set to  $n$  times of its response-time for one-round execution. In the following discussion, we assumed that process diagrams of *TCS* are acyclic.

Step two: *sub-TCSs* identification. The *sub-TCS* identification algorithm is stated in figure 3. Figure 4 shows an example of *sub-TCS* identification for a *TCS* whose process includes selection and parallel structure. The number on each task is its average response-time. (The definition of average response-time will be given later in this article.)

Step three: *MAT* and *SL* setting. We just set *sub-TCSs* the same *SL* as the *TCS* that it belongs to. But it is more complex to set their *MAT*. It is accomplished by 4 steps.

1. Inquiring of the service broker about average response-time of all tasks in a *TCS*. The average response-time of a task is the average of all of its service instance's response-time recently. The article adopts JBCLMS [3] as the service broker that developed by

Peking University. The formula for calculating average response-time is as follow:

$$\bar{r}(t_i) = \frac{\sum_{j=1}^n \bar{r}(s_{ij})}{n} \quad (1)$$

Where  $\bar{r}(t_i)$  is the average response-time of task  $t_i$ ,  $s_{ij}$  is a service instance of  $t_i$ ,  $\bar{r}(s_{ij})$  is the average response-time of  $s_{ij}$  (i.e. the average of several recent measuring values of  $s_{ij}$ 's response-time).  $n$  is the amount service instances of task  $t_i$ .

$s \leftarrow$  The TCS in which sub-TCS will be identified  
*SSIidentification(s, FALSE);*

```

function SSIidentification(in ProcSec, Out SSIidentified)
//ProcSec is a process section whose sub-TCSs need to be identified
//SSIidentified is a Boolean variable. It is FALSE by default, and TRUE When a
//new sub-TCS is identified,
{ BEGINNODE←begin node of the process in ProcSec;
  ENDNODE←end node of the process in ProcSec;
  NODE←BEGINNODE;
  SSIidentified←FALSE;
  while (NODE!=ENDNODE){
    NODE←next node of the process in ProcSec;
    switch NODE is type of {
      case conditional node:
        for(each conditional branch){
          SSIidentified=TRUE;
          NewProcSec←this conditional branch;
          SSIidentification(NewProcSec,FLAG);
          if(!FLAG) mark NewProcSec as a sub-TCS; }
        case fork node:
          for(each concurrent thread){
            SSIidentified=TRUE;
            NewProcSec←this thread;
            SSIidentification(NewProcSec,FLAG);
            if(!FLAG) mark NewProcSec as a sub-TCS; }
          } //switch end
    } //while end
  } //function end

```

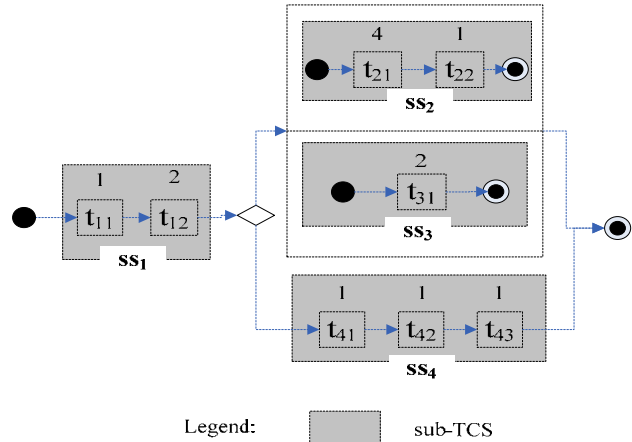
**Figure 3. Sub-TCS identification algorithm**

2. Calculating average execution time of each sub-TCS. The average execution time of sub-TCS  $ss_k$  is denoted as  $\bar{e}(ss_k)$ , and the formula for its calculation is as follow:

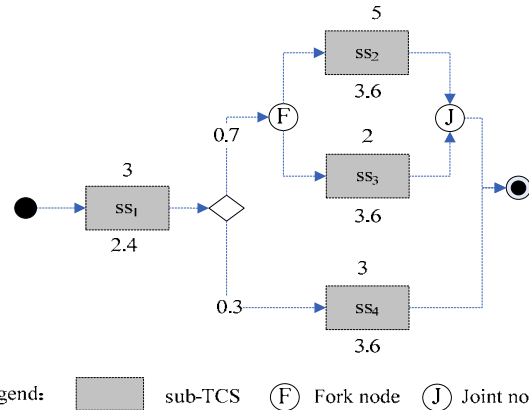
$$\bar{e}(ss_k) = \sum_{l=1}^m \bar{r}(t_{kl}) \quad (2)$$

Where  $t_{k1}, t_{k2} \dots t_{km}$  are all tasks in  $ss_k$ .

3. Generating sub-TCS diagram. Sub-TCS diagram is an extended net-work diagram[4], where all sub-TCSs are identified as nodes, the average execution time of sub-TCSs are weight of nodes, and the transitions between sub-TCSs are dependences between nodes. We add selection nodes into the diagram to connect different condition branches and assign execution probability for each branch (the probability can be gotten through the statistic of the composite service's historical data of execution, or it can be simply set by domain experts). In addition, we add fork and joint nodes to illustrate concurrent structures. Figure 5 shows the sub-TCS diagram of the TCS in figure 4. The number on edges represents execution probabilities of condition branches. The values above nodes are their weights.



**Fig.4 An example for sub-TCS identification in a TCS**



Note: the numbers above nodes are average response-time of sub-TCSs, the numbers under nodes are maximum-acceptable-time, the TCS's maximum-acceptable-time is 6 seconds.

**Figure 5. An example for assigning MAT for sub-TCSs in a TCS**

4. Calculating the MAT of sub-TCSs. The algorithm for calculating sub-TCSs' MAT is showed in figure 6. The algorithm contains two functions. Function *ComputeCriticalPathLen* is used to calculate the length of critical path in sub-TCS diagram. Here, the length of a path refers to the sum of all nodes' weight the path passes by. The length of critical path is the biggest length of all paths from the initial node to the final node. In the calculation of critical-path's length, as for multi-branches that connected by the same conditional selection node, we regard the probability on each branch as the branch's weight, and calculate weighted average valued of path length of all branches. We use the average value as the path length of all the condition branches. As for those concurrent threads, we use the longest length of all these threads as the path length for all of them. Another function *ComputeMAT* calculates MAT of each sub-TCS by using the length of critical path returned by function *ComputeCriticalPathLen*. For the nodes (sub-TCSs) in a critical path, the algorithm makes the sum of their MAT equals to the MAT of the TCS, and their MAT is



proportional to its weight. For those nodes that are not in a critical path, their *MAT* are longer than the nodes in critical paths with the same weight, and the sum of *MAT* of nodes in any paths from the initial node to the final node equal to the *TCS*'s *MAT*.

### 3.3 Run-time service-instances selection

The identification and decomposition of all *TCSs* in a composite web service are performed before the composite service's execution. During the execution of composite services, service broker will select service instances that possess appropriate response-time for each task according to *MAT* and *SL* of each *sub-TCS*. The run-time service-instance selection process includes the following three steps.

Step one: service execution engine report information of *sub-TCS* to service broker. While a *sub-TCS* start to execute (i.e. the first task of the *sub-TCS* is about to execute), the service execution engine that responsible for parsing and performing composite service will submit a request to a related service broker and report run-time information of the *sub-TCS* to the service broker. The information includes the structure of a *sub-TCS* (i.e. what tasks are included in the *sub-TCS* and their execution sequence), the *MAT* and *SL* of the *sub-TCS*.

Step two: Service broker reserve *MAT* for *sub-TCS* according to its *SL* on receiving the report from a service execution engine. The changing of service instances' response-time and accessibility during the execution of a composite web service will result in lateness risks of *TCSs* and *sub-TCSs*. In order to reduce the lateness risks for those *sub-TCSs* which have higher *SLs*, service broker will reserve some time from their *MAT*, that is to say, the service broker will deflate their *MAT* reported by the service execution engine, and use the deflated ones in service instance selection. The deflating ratio is decided by the *sub-TCS*'s *SL*. The higher a *sub-TCS*'s *SL* is, the more *MAT* will be reserved. We denote the deflating ratio for *sub-TCSs* whose *SLs* are 0, 1, ..., *n* as  $r_0, r_1, \dots, r_n$  ( $0 \leq r_0 \leq r_1 \leq \dots \leq r_n < 1$ ).

As for *sub-TCS ss*, we notified its *MAT* that is reported by its service execution engine as  $nt(ss)$ , and the deflated *MAT* as  $st(ss)$ . The formula for calculating  $nt(ss)$  is as follow:

$$st(ss) = (1 - r_i) \times nt(ss) \quad (3)$$

After reporting information of a *sub-TCS* to service broker, the service execution engine starts performing the first task in the *sub-TCS*. Every time when a task is about to perform, the service execution engine submits a service-instance-selection-request to the service broke. On receiving this request, the run-time service-instance selection process comes to step three, service broker selects response-time of each task in *sub-TCS* dynamically.

Step three: This step can be divided into three sequential sub-steps:

1) Service broker calculates target response-time for the task which is about to execute. In response-time

management we always try to make a task's actual response-time equal to its target response-time. As for the *sub-TCS ss*, we denote the *i*th task in its task sequence as  $t_i^{ss}$ , and we denote  $t_i^{ss}$ 's target response-time as  $o(t_i^{ss})$ . The formula for calculating  $o(t_i^{ss})$  is as follow:

$$o(t_i^{ss}) = \frac{\bar{r}(t_i^{ss})}{r(ss) - (T - T_0)} \times [st(ss) - (T - T_0)] \quad (4)$$

Where  $T_0$  is the starting time of the *ss* (i.e. the time that service execution engine report the information of *ss* to the service broker),  $T$  is the current time. Since tasks in a *sub-TCS* are performed sequentially, while the request of task *i* is arriving, the former *i-1* tasks have been finished already. The total execution time for the former *i-1* tasks is known as  $T - T_0$ . The above formula uses this information to adjust a task's target response-time dynamically. That is when the value of  $T - T_0$  is bigger than the sum of objective response-time of the former *i-1* tasks, which means the former tasks execute too slowly, we will try to catch up by selecting faster service instances. On the contrary, when the value of  $T - T_0$  is smaller than the sum of objective response-time of the former *i-1* tasks, we can select service instances which are slower but better in other QoS attributes, for example a slower but cheaper service instance.

2) Service broker selects service instance for the task. The selection principle is that if there exists at least one service instance that its response-time is less than target scheduling time, and then we select the one that process other best QoS attributes, for example, the service which has the cheapest price or the highest reliability. Otherwise we select service instance which has the shortest response-time.

3) Service broker notifies the selected service instances to composite service execution engine, and thus the service instance is invoked.

```

ss←sub-TCS diagram of a TCS whose sub-TCS's MAT need to be calculated;
t←MAT of theTCS;
ComputeMAT(ss, t);

```

```

function ComputeCriticalPathLen(in ProcSec, in P
Out CPL)
//ProcSec: fragment of a sub-TCS diagram
//P: execution probability of ProcSec
//CPL: Length of the critical path in ProcSec
{ BEGINNODE←begin node of the process in ProcSec;
  ENDNODE←end node of the process in ProcSec;
  NODE←BEGINNODE;
  while (NODE≠ENDNODE) {
    NODE←next node of the process in ProcSec;
    switch NODE is type of do {
    case sub-TCS node:
      //NODE.ET: the average execution time of NODE
      CPL ←CPL+P*NODE.ET;
    case conditional node:
      for(each conditional branch){
        NewProcSec←this conditional branch;
        //NewProcSec.P: execution probability of NewProcSec
        //CBT: temporary variables
        ComputeCriticalPathLen(NewProcSec,NewProcSec.P, CBT);
        CPL ←CPL+CBT*P;
      }//for end
    case fork node:
      for(each concurrent thread){
        NewProcSec←this concurrent thread;
        //TEDT: temporary array, TEDT[i] is the average
        //execution time of concurrent thread i
        ComputeCriticalPathLen(NewProcSec,P,TEDT[i]);
      } //for end
      //Max is a function returns the maximum value of each element
      //in array TEDT
      CPL ←CPL+Max(TEDT)
    } //switch end
  } //while end
} //function end

```

```

function ComputeMAT(in ProcSec, in MAT)
// ProcSec: fragment of a sub-TCS diagram
// MAT: the MAT of ProcSec
{ BEGINNODE←begin node of the process in ProcSec;
  ENDNODE←end node of the process in ProcSec;
  NODE←BEGINNODE;
  //call ComputeCriticalPathLen to calculate the critical-path-length of ProcSec
  ComputeCriticalPathLen(ProcSec, 1, CPT);
  while (NODE≠ENDNODE) {
    NODE←next node of the process in ProcSec;
    switch NODE is type of do {
    case sub-TCS node:
      // NODE.MAT: the MAT of NODE
      //NODE.ET: the average execution time of NODE
      NODE.MAT←(NODE.ET/CPT)*MAT;
    case conditional node:
      // CBET: temporary variables
      CBET←0;
      for(each conditional branch){
        NewProcSec←this conditional branch;
        //NewProcSec.P: the execution probability of NewProcSec
        ComputeCriticalPathLen(NewProcSec, NewProcSec.P, CBET);
        CBET ←CBET+CBT; } //for end
      MaxCB←(CBET/CPT)*MAT;
      for(each conditional branch){
        NewProcSec←this conditional branch;
        ComputeMAT(NewProcSec, MaxCB);}
    case fork node:
      for(each concurrent thread) {
        NewProcSec←this concurrent thread;
        // TEDT: TEDT:array,where TEDT[i] store the average
        //execution time of concurrent thread i
        ComputeCriticalPathLen(NewProcSec, NewProcSec.P, TEDT[i]);}
      //Max is a function returns the maximum value of each element
      //in array TEDT
      MaxTHD←(Max(TEDT)/CPT)*MAT
      for(each conditional branch){
        NewProcSec←this concurrent thread;
        ComputeMAT(NewProcSec, MaxTHD); }
    } //switch end
  } //while end
} //function end

```

Figure 6. MAT calculation algorithm for sub-TCSs

## 4. Experimentation

In order to validate the method proposed in this article, we compiled a composite service that includes one single TCS as shown in figure 4. The experiment adopts JBCLMS that developed by Peking University as UDDI registry and service broker [3] [4]. All PCs that carry the experiment had the same configuration: Pentium IV 2GHz with 512M RAM. The reserving ratio of MAT for TCS and sub-TCS with SL 1, 2 and 3 are 0, 0.1 and 0.2 respectively. Every task has 5 available service instances, and the average response-time of each service instance is shown in table 1. We set an assumption that those service instances have longer response-time are better in other QoS attributes. This assumption makes our method selects service instances that have the longest response-time among those instances whose response-time are less than the task's objective response-time.

The experiment is carried out under two conditions:

Condition A: All service instances' response-time remain unchanged during the execution of the composite web service

Condition B: The response-time of each service instance floats in the range of 20 to 40 percent with a 50

percents probability in every 0.5 seconds, and every service instance turns into un-accessible for 2 seconds with a 30 percents probability in every 6 seconds.

Table 1. Average response-time of service instances in the experiment

task	average response-time for service instances (unit:sec)				
	1	2	3	4	5
$t_{11}$	0.8	0.9	1.0	1.1	1.2
$t_{12}$	1.6	1.8	2.0	2.2	2.4
$t_{21}$	2.4	3.2	4.0	4.8	5.6
$t_{22}$	0.8	0.9	1.0	1.1	1.2
$t_{31}$	1.4	1.7	2.0	2.3	2.6
$t_{41}$	0.7	0.9	1.0	1.1	1.3
$t_{42}$	0.3	0.5	1.0	1.5	1.7
$t_{43}$	0.2	0.4	1.0	1.2	2.2

Under these two conditions, we adjust the MAT and the SL of the TCS. Each case is carried out independently for 10 times. And we adopt the average value as the result of the experiment. The experiment data is shown in figure 7.

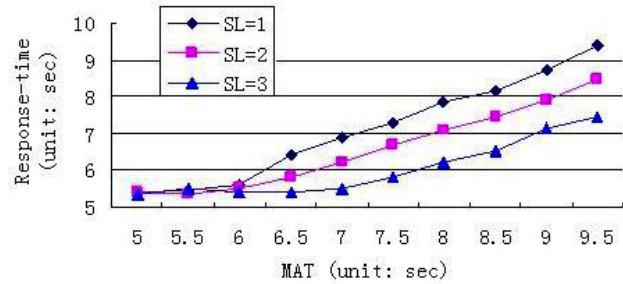
From figure 7.a and 7.b we can see that under condition A, when the *MAT* is less than 6 seconds, the actual response-time of *TCS* remains stability with different *SLs*. The reason is that even all tasks are bound to the fastest service instances, the *TCS* need 5.6 second to response, so when the *MAT* is near or less than this utmost time, the method always selects the fastest service instances for tasks on the critical path regardless of the *SLs*. When the *MAT* is greater than 6 seconds, the *TCS*'s response-time increases along with the increasing of the *MAT* but never exceeds its *MAT*, and because of the *MAT* reservation the *TCSs* with higher *SLs* have a bigger difference between their response-time and *MAT*. This shows that in a environment where all service instances' response-time is stable(i.e. condition A), the mechanism of *MAT* reservation is not necessary. But if the service instances' response-time changes frequently, e.g. condition B, which is more like the actual environment, things are different. From figure 7.c and figure 7.d we can tell random floating of *TCSs*' response-time under condition B. Especially In figure 7.d, when the *MAT* is less than 7 seconds and the *SL* is low, *TCSs*' response-time exceeds its *MAT*. But under the same circumstances, when the *SL* is higher, the *MAT* reservation mechanism makes the probability of service lateness decline obviously. This means that setting higher *SL* for sensitive *TCSs* can effectively reduce their lateness risk.

## 5. Related works

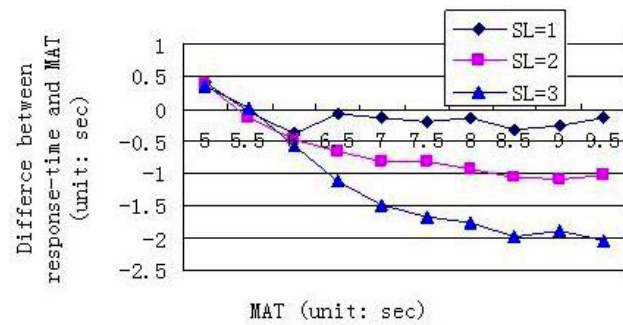
[6][7] studied the optimization and management of composite web services' QoS which included response-time and price, and selected service instances that had proper QoS. [8] gave weight to different QoS attributes according to users' preference, and then it put forward a normalization method to make different QoS attributes' value into an overall value according their weight, thus it facilitated the selection of proper service instances. The methods mentioned above didn't consider the relationship among tasks while selecting service instances for tasks in composite service. [1][2][9][10] considered the relationships between tasks in composite service and adopted methods such as linear programming, genetic algorithms to perform a global optimization of QoS in the range of all tasks in a composite web service. Differing with the above works, the method proposed in this article performs a simple optimization of service's response-time neither in the range of a single task nor all tasks in a composite service. Instead, our method adopts a compromising solution in which the optimization is performed in the range of *TCSs* which contains subsets of all tasks in a composite service. *TCSs* provide a mechanism to combine users' perspectives of a composite service to providers' perspectives in response-time management. By using *TCSs* and its relative *MAT* and *SL*, service provider can manage a composite service's response-time according to its users' feelings, and thus increases the degree of users' satisfaction.

In order to cope with the changing of service instances' QoS during the execution of a composite service, [1] [2] [9][10] re-executed their QoS optimization

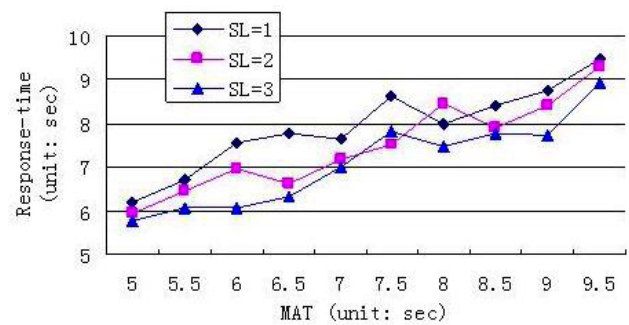
algorithms in the range of un-performed tasks to replan the composite service's execution at run-time. However, the inheritance algorithm and linear programming adopted in their replanning methods are too complex, and it makes the real-time requirements for them are hard to meet with. In our method most calculation were finished before composite services' execution, and the left work that has to be done at run-time is simple. Our run-time replanning algorithm simply adjusts later tasks' speed according to the sum of finished tasks' response-time and the *MAT* of the current *TCS* or *sub-TCS*. The simple run-time replanning algorithm makes our method more practicable.



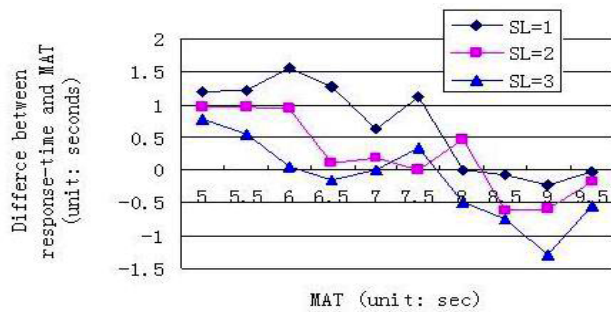
(a) Trends of response-time of *TCSs* with different *SLs* under condition A



(b) Trends of the difference between response-time and *MAT* of *TCSs* with different *SLs* under condition A



(c) Trends of response-time of *TCSs* with different *SLs* under condition B



(D) Trends of the difference between response-time and MAT of TCSs with different SLs under condition B

**Figure 7. Response-time of TCSs with different MAT and SL under two different conditions**

## 6. Conclusion

Web service QoS includes many attributes such as response-time, price, successful implementation rate, accessibility etc [11][12], where response-time is one of the most important attributes that is studied by many researchers because it is the key problem when design and manage business process[13][14]. The structures like circle, parallel, condition selection make the calculation of response-time more complicated than other QoS attributes. The article studied composite web services' response-time and proposed a using scenario oriented response-time management method. This method identified TCSs according to *waiting cycles* in the interactions between users and composite web services, and assigned deferent SLs to TCSs to identify their different weight on the effect of user's satisfaction. TCSs provide a mechanism to combine users' perspectives of a composite service to providers' perspectives in response-time management. By utilizing TCSs not only the total response-time of composite services but also *waiting cycles* are managed, and thus increased users' satisfaction for composite services. Then the method performed a simple optimization of service's response-time in the range of TCSs, and applied a simple and fast service instances selection and dynamic replanning algorithm for composite web services at run-time. The simplism of the run-time algorithms made our method more practicable. The experiment validated the efficiency of the proposed method. The future work of the article is to extend the method to multiple QoS attributes like reliability, usability, price and so on.

## References

- [1] Zeng L.Z., Benatallah B. and Dumas M., "QoS-Aware Middleware for Web Services Composition", IEEE Transaction on Software Engineering, 2004, 30(5):311~327
- [2] Canfora G., Di Penta M., Esposito R., Villani M.L., "Qos-Aware Replanning of Composite Web Services", Proceeding of the IEEE International Conference on Web

Services(ICWS'05), Orlando, Florida USA, 2005, 121~129

- [3] Zhao J.F., "Research on feedback management and run-time application supporting techniques for software component library", PhD thesis, Peking University, Peking PRC, 2005(in Chinese)
- [4] Shao L.S., Li T., Zhao J.F., Wang Y.S., Xie B., Mei H., An Extensible Management Framework for Web Service QoS, Chinese Journal of Computers, 2008, Vol. 31, No.6:1458~1471
- [5] Pinedof M., Scheduling: Theory, Algorithms, and Systems, second ed. New York, USA: Prentice Hall, 2001
- [6] Georgakopoulos D., Schuster H., Cichocki A., Baker D., "Managing process and service fusion in virtual enterprises", Information System, special issue on Information System Support for Electronic Commerce, 1999, 24(6):429~456
- [7] Casati F., Shan MC., "Dynamic and adaptive composition of e-services", Information Systems, 2001, 26(3):43~162
- [8] Liu Y.T., Ngu H.H. Anne, Zeng L.Z.. QoS computation and policing in dynamic web service selection. In: Proceeding of World Wide Web Conference(WWW'04), New York, New York, USA, 2005, 66~73
- [9] Cardoso J., "Quality of service and semantic composition of workflows", PhD thesis, University of Georgia, Athens, Georgia, USA, 2002
- [10] Ardagna D., Pernici B., "Global and local QoS constraints guarantee in web service selection", Proceeding of the IEEE International Conference on Web Services(ICWS'05), Orlando, Florida USA, 2005, 806~807
- [11] Ran S.P., "A model for web services discovery with QoS", ACM SIGecom Exchanges, 2003, 4(1):1~10.
- [12] Zhao J.F., Wang Y.S., Xie B., Yang F.Q., "A management framework of component supporting QoS of component", ACTA Electronica Sinica, 2004, 32(12A):165~168(in Chinese)
- [13] Eder J., Panagos E., Rabinovich M., "Time constraints in workflow systems", Advanced Information Systems Engineering: 11th International Conference(CAiSE'99), Springer-Verlag, Heidelberg, Germany, 1999, 286~300
- [14] Gillmann M., Weikum G., Wonner, W., "Workflow management with service quality guarantees", Proceeding of ACM SIGMOD International Conference, Management of Data, Madison, USA, 2002, 228~239

## ACKNOWLEDGEMENT

This research was sponsored by the National Grand Fundamental Research 973 Program (SN: 2005CB321805), the High-Tech Research and Development Program of China (SN: 2007AA010301), the Science Fund for Creative Research Groups of China (SN: 60821003), and the National Nature Science Foundation (SN: 60803011) in China.

# Dynamic Service Composition for Virtual UPnP Device Creation

Sheng-Tzong Cheng<sup>1</sup>, Chih-Lun Chou<sup>1</sup>, Jiashing Shih<sup>1</sup>, Mingzoo Wu<sup>2</sup>

<sup>1</sup>Dept. of Computer Science and Information Engineering, National Cheng Kung University, Tainan,

<sup>2</sup>IDEAS, Institute for Information Industry, Kaohsiung,

<sup>1,2</sup>Taiwan, R.O.C.

stcheng@mail.ncku.edu.tw, {mikechou, jason}@csie.ncku.edu.tw, wu.mingzoo@gmail.com

## Abstract

UPnP devices and services lack a composition framework to provide a novel value-added service. This paper aims at designing and implementing a dynamic service composition framework and creating a virtual UPnP device in a home network environment. Semantic data type ontology is used to define a communication interface for UPnP services. The interface matching mechanism is employed to construct a service graph that describes which services can be composed together. Finally, the proposed system travels on the service graph, and a method called Virtual Application Probing which allows home virtual applications to be dynamically and semantically composed from the individual services of home networked devices and find a suitable execution path to generate a new device. Home users can invoke this newly generated device through the control point, as if the device is real in the house. In addition, a virtual UPnP Karaoke device generated by the proposed service composition system is demonstrated as well. From the demonstration, it can be seen that the service composition system is feasible in practice.

**Keyword:** UPnP, semantic data type ontology, service composition, virtual device creation

## 1 Introduction

With the proliferation of home networked devices, all sorts of devices could be discovered and controlled by UPnP protocols [1]. UPnP uses common protocols which are independent of the underlying physical media and transports, and ensure every device vendor could follow. Many industry companies and research initiatives such as Universal Plug and Play (UPnP), Open Services Gateway Initiative (OSGi) [2], Digital Living Network Alliance (DLNA) [3] and Home Audio and Video Interoperability (HAVi) [4] have tried to understand the communication protocol between control point and devices. However, up to now, they are very little to get in touch with composing the primitive services to create complex value-added services.

UPnP device description includes the vendor-specific,

manufacturer information, and URLs to vendor-specific Web sites, etc. For each service included in the device, the device description lists the service type, name, a URL for a service description, a URL for control, and a URL for eventing. UPnP service description is provided by a UPnP vendor. The description is recorded in XML-based syntax and is usually based on a standard UPnP service template. A UPnP service template is produced by a UPnP Forum [11] working committee. And UPnP Forum working committees have defined standard device architecture for UPnP vendors to follow so as to build their intelligent devices. UPnP is an open networking architecture that uses Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices at home, office, and public spaces.

UPnP protocol includes addressing, discovery, description, control, event notification, and presentation. The foundation for UPnP networking is IP addressing. Each device must have an IP address, which can be obtained from a Dynamic Host Configuration Protocol (DHCP) server or generated from Auto-IP configuration. Given an IP address, the first step in UPnP networking is discovery. Control point can search for the interesting device on the UPnP network. Besides, a UPnP device can advertise its services to control points on the network. The UPnP discovery protocol is based on the Simple Service Discovery Protocol (SSDP).

In the control step, the control point may send actions to device's service. The control URL in the device description is where control messages are sent. Control messages are expressed in XML using Simple Object Access Protocol (SOAP). Service returns the action results to control point.

To get events, control points subscribe to eventing for a specific service within a device. Then when the service has an event, it sends that event to all current subscribers. UPnP uses General Event Notification Architecture (GENA) to define subscriptions and event notifications. The final step in UPnP networking is presentation. A URL for presentation page is given in the device description. In summary, UPnP is a Web-based communication protocols between control point and devices.

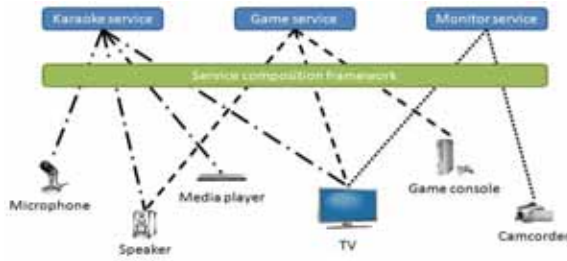


Fig. 1: The concept of service composition

Web services use Universal Description, Discovery, and Integration (UDDI) for discovery, Web Services Description Languages (WSDL) for description and Simple Object Access Protocol (SOAP) for communication. As Web services become more and more prevalent, many Web service composition technologies are proposed and developed. Business Process Execution Language for Web Services (BPEL4WS) [5] developed by IBM, Microsoft, SAP and Siebel supports process-oriented service composition. It only represents a specific process composition flow and dynamic service composition is not supported. So there are many research projects propose dynamic service composition technologies. CoSMoS [6] presents a dynamic service composition system using semantic information. CoSMoS can integrate the services to construct a semantic graph. Given a user request, CoSMoS can check the semantics and generate an execution path. A template-based composition system is proposed in eFlow [7], in which a composite service is designed by a template which defines an order of execution of the services. User can choose an application template from the repository or by creating a template by himself. The request application is composed through selecting the services specified in the template and combining them according to the structure described in the template. Users can replace the default services with the one that best suit their needs. There are some other works [8][9] focused on service selection issues. Dynamic service selection is an important issue in Web service composition. Since the Web environment changes frequently, there are many similar services for users to choose so that the total quality is the best. For example, users may choose the best suit service depending on the network traffic, server loading etc.

In this paper, we aim at designing and implementing a dynamic service composition system and create a virtual device in home network environment. We propose data type ontology to define a communication interface for a service. And we show how to use semantic descriptions to aid in the dynamic service composition system. Also, we present a method called Virtual Application Probing (VAP) which allows home virtual applications can be dynamically and semantically composed from the individual services of home networked devices. We design and implement the dynamic service composition system using existing

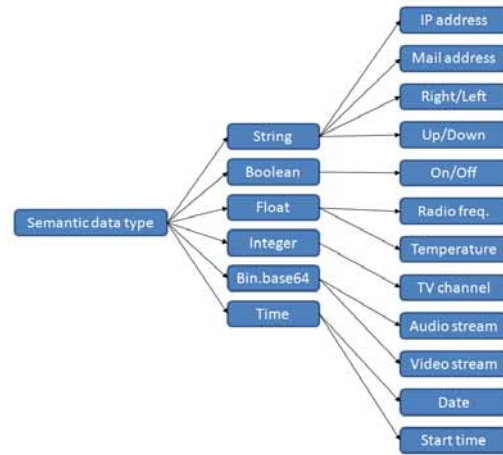


Fig. 2: Semantic data type ontology

technologies such as UPnP, ontology, and XML. Then our system could create a virtual device that was composed from the primitive devices in home network environment.

The rest of this paper is organized as follows. Sec. 2 describes the design for our virtual device creation system. In Sec. 3, we describe the implementation technique. Sec. 4 presents the demonstration of the proposed system. Finally, conclusion remarks are drawn in Sec. 5.

## 2 System Architecture

Fig. 1 shows the concept of dynamic service composition. Through our dynamic service composition system, these useful services could cooperate with each other to create a virtual device or a novel application. For example, virtual Karaoke device is composed from microphone, speaker, media player and TV. When home users invoke the virtual Karaoke device from control point then these primitive devices would be invoked automatically. It means that the TV would play a music video from media player, and the microphone would stream the voice to speaker automatically.

### 2.1 Data Type Ontology Classification

UPnP forum working committees have defined some data types for action variables. But in our view point, only data type is not sufficient for dynamic service composition. So we extend the service description and add semantics information to define a service interface. The I/O variables are classified according to data type at first, such as Bin.base64, String, Float, Integer, Time, Number, Boolean, etc. These data types are further classified according to semantics. For instance, the data type Bin.base64 is the data format for media transformation and could be subdivided into video stream, audio stream, text, picture, etc. The data type Float has the semantics, such as target temperature for air-conditioner or frequency for radio channel.

The data type Integer may have the semantic

representations, such as channel for a TV program. UPnP control point may retrieve device and service information with semantics to provide accessible services. If the data type ontology share and publish the same underlying ontology of the used terms, then control point can extract and aggregate the data type and semantic information to do service matching. The benefit of using data type ontology is that it is easy for developers to design service interfaces and is easy for users to understand. After defining the data type ontology, we use the data type and semantic information to describe the service interface. The followings introduce UPnP home networked device designed with semantics descriptions. Every service has input interface and output interface. With interface matching method, we could know whether the service's output can be fed into the next service's input. Fig. 2 is the data type ontology diagram of the variables for communication interface of home networked devices. And it shows that the variables are first classified by the data types, and then classified by semantics.

## 2.2 Service Interface Matching

UPnP service description defines actions, arguments, state variables, data type, range, and event characteristics. A service interface specifies methods that can be performed on the service. Service's interfaces are public for an external use. A service can hold two sorts of interfaces: input interface and output interface. Traditionally, UPnP service description only has data type and input/output information to describe a service interface. In our view point, it is not enough to perform dynamic service composition. For example, the data type of microphone's output interface is the same as the printer's input interface. Only with data type information, we might think that microphone's output could be fed to printer's input. But the output of a microphone is an audio stream and the input of a printer is a text file. Although their data types are the same, but microphone's output cannot be fed to printer's input.

With the support of data type ontology, semantics of a service can be freely defined, thus providing high extensibility. The matching of two services is using the information in the service interface. If service A's output interface is exactly the same with service B's input interface. Then service A and service B can be composed together.

## 2.3 Create Semantic Interface

After defining the data type ontology, we use the data type and semantic information to describe the service interfaces. Every service has two interfaces, one is input interface and the other is output interface. With interface matching method, we could know whether the service's output can be fed into the next service's input.

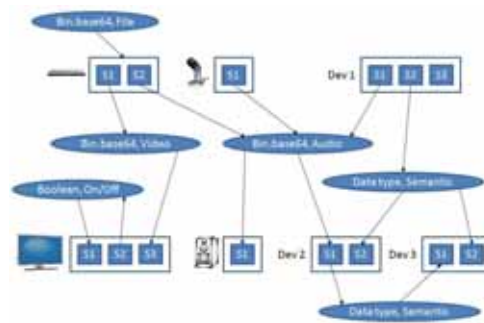


Fig. 3: Directed service graph

Suppose that a TV has four services: SetPower, GetPower, SetChannel, and Visual. SetPower service can switch the TV to power on or power off. SetChannel service can select TV programs. And Visual service can play a movie from media player or display a TV program. The input interface of SetPower has two variables, its data type is Boolean and its semantic is On/Off. Device\_Status is the data type and TV is the semantic for SetPower's output interface. The input interface of SetChannel has two variables, its data type is Integer and semantic is Channel. Device\_Status is the data type and TV is the semantic for SetChannel's output interface. The input interface of Visual service has two variables; its data type is Bin.base64 and its semantic is Video. Device\_Status is the data type and TV is the semantic for output interface.

## 2.4 Service Graph Construction

Service graph is an intuitive way to represent composition concepts. Two types of nodes are defined in a service graph: Data Semantic Node (DSN) and Service Information Node (SIN). The links between these nodes represents their associations. The service graph represents the composition information.

- **DSN:** with the aid of data type ontology, we can create several kinds of DSN. Each DSN represents a pair of data type and semantic defined by the data type ontology to represents the service's interface. In the later discussion, we briefly write (Data type, Semantic) to represent DSN.
- **SIN:** Every device may have many services and each service has input and output interfaces. We take down the device and service information in the SIN, which has two kinds of information, the device name and the service name. We simply write (Device name, Service name) for later discussion.

When a UPnP device is discovered, our system first check the service interface's data type and semantic. If the DSN's data type and semantics are the same with the service input interface's data type and semantics, then a link is directed from DSN to SIN. Otherwise, if the DSN's data type and semantic are the same with the service output interface's data type and semantics, then a link is directed from SIN to DSN. Each time when a device is discovered

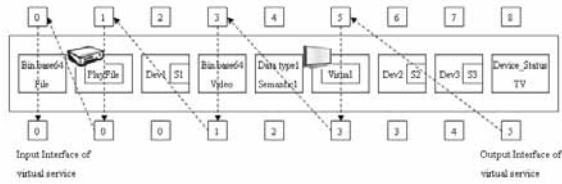


Fig. 4: Exploration of execution path

by UPnP control point in home networks, our system would make links between DSN and SIN in service graph automatically. Fig. 3 shows a directed service graph. After constructing the directed service graph, we can find an execution path of virtual device which is composed from the primitive services.

## 2.5 Execution Path Exploration

We use Virtual Application Probing (VAP) to find a composite service path in the directed service graph. VAP is somewhat similar to Breadth First Search with extra index information which records the preceding node. To implement this scheme, we place each visited node into a linked list and record its preceding node with index information. Then we use VAP to find a shortest execution path in service graph. The followings describe the procedures of the VAP.

**Step 0:** Check virtual device description. Before finding an execution path, control point would read the virtual device's description to know the virtual service's input and output interfaces' data types and semantics. After knowing the DSN of the input and output interfaces, we would find a shortest composite execution path between the two DSNs.

**Step 1:** Put the DSN which represents the virtual service's input interface into linked list.

**Step 2:** Put the SIN whose input interface is the DSN from step 1 into linked list. And record their indexes as the number of the preceding nodes in the linked list.

**Step 3:** Put the DSN which represents the output interface of the SIN from step 2 into list. And record the index as the number of the predecessor nodes in the linked list.

**Step 4:** If we find DSN of the virtual service's output interface, stop searching and export the execution path according to their indexes. Otherwise, repeat Step 2 and Step 3 until there are no nodes left in linked list. Notice that a node would not be visited twice in searching procedure.

Once we detect that the output interface of virtual service, then stop probing and export the path according to the indexes from the linked list. Fig. 4 describes how to export the execution path from the linked list. Referring to the index recorded by (Device\_Status, TV), we can find (TV Device, Visual). And so forth, by the index of (TV Device, Visual) we find (Bin.base64, Video). By the index of (Bin.base64, Video) we find (Player Device, PlayFile). And by the index of (Player Device, PlayFile) we find (Bin.base64, File). At last the execution path comes out:

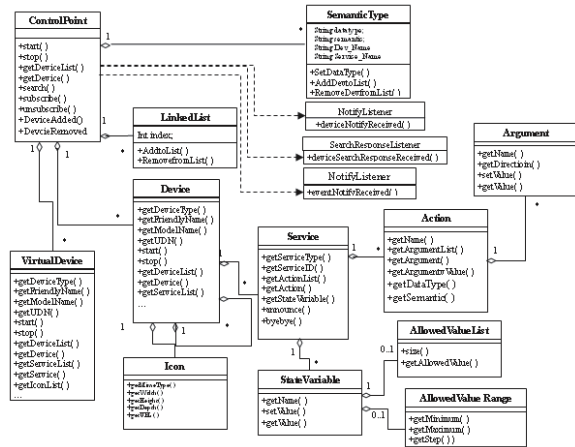


Fig. 5: Class overview of service composition system

(Bin.base64, File) → (Player Device, PlayFile) → (bin.base64, Video) → (TV Device, Visual) → (Device\_Status, TV).

## 2.6 Virtual Service Provision

Once all execution paths of virtual device are explored, the virtual device with the execution paths will be created and control point would invoke the services which was required in the execution path sequentially. We describe two types of flows: control flow and data flow. Control flow is used to trigger the required service in execution path. Then the control point has the ability to integrate two primitive services to establish a direct data flow.

## 3 System Implementation

Our device model is refined from CyberLink for UPnP [10]. CyberLink controls these protocols automatically, and supports to create your devices and control points quickly. We append some classes and methods to implement our dynamic service composition system. Fig. 5 shows that we added DeviceAdded(), DeviceRemoved(), getDateType(), getSemantic() methods and VirtualDevice class, SemanticType class and LinkedList class from the class diagram of CyberLink for UPnP. The implemented classes are described as follows.

- **ControlPoint Class:** We can create an instance of ControlPoint Class to create a UPnP control point. Use ControlPoint::start() to activate the control point. The control point multicasts a discovery message searching for all devices to the UPnP network automatically when the control point is active. The control point can send action or query control messages to the discovered devices. To send the action control message, use Action::setArgumentValue() and Action::postControlAction(). In ControlPoint class, we implement DeviceAdded() and DeviceRemoved() methods. UPnP control point receives notify events from devices in the UPnP network, and the devices are added or removed



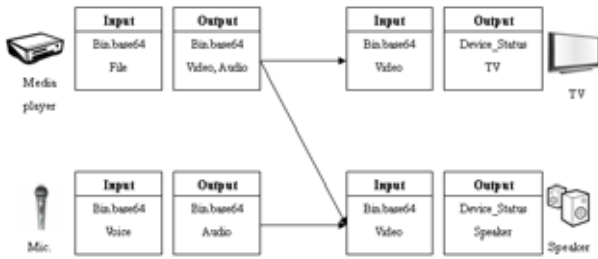


Fig. 6: Creation of virtual Karaoke device

```

Filename: Speaker/description/service/power/description.xml
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    ...
  </specVersion>
  <actionList>
    <action>
      <datatype>Bin.base64</datatype>
      <semantic>Audio</semantic>
      <direction>in</direction>
    </action>
  </actionList>
</scpd>

```

Fig. 7: Semantic service description

from the control point automatically. The expired device is added or removed from the device list of the control point automatically. When a UPnP device is discovered by control point, DeviceAdded() would check whether the service can be composed with other service by interface matching. If the added service's interface exactly matches another service's interface then we connect the two service nodes in service graph. On the other hand, if a UPnP device is leaving the UPnP network, DeviceRemoved() removes the links between the service nodes in service graph. We implement DeviceAdded() and DeviceRemoved() methods to maintain the service graph.

- **SemanticType Class:** This class is used to describe the data structures of service graph. In SemanticType domain, we define the two nodes in service graph. One is DSN and the other is SIN. DSN represent the interfaces of a service. And SIN records the service information including device name and service name. SemanticType class defines data type and semantics to represents services' interface. For example, Audio, Video and Text can be defined as semantics. With the aid of semantics, we use "interface matching" approach to find out what service's output can be fed to the next service's input. Then we use VAP to find a virtual service in the directed service graph.
- **LinkedList Class:** The LinkedList class is designed to find execution path of virtual device. We implement a linked list to record the service graph nodes in execution path. And extra index information is used to record the preceding node's number in the linked list. Section 2.5 describes how to find an execution path using a linked list and index information in detail before.
- **VirtualDevice Class:** Class VirtualDevice is a general virtual device class inherited from ControlPoint. In the beginning of VAP process, control point will check the

virtual device's descriptions first. If the virtual service can be composed from the primitive services, we create an instance of virtual device class with the execution path and user can invoke the virtual device from control point. The main difference between real device and virtual device is that the virtual service is not implemented in advance. A real service for real device performs a specific task designed by inventor in advance. For example, the Visual service for TV is designed to play video stream or a TV program. Unlike the real device, the virtual service/device is not implemented beforehand. The virtual device can receive a composite execution path found by VAP. The virtual service is a composite service composed from the primitive services. Once the virtual service is invoked, control point would sequentially invoke the primitive services involved in execution path.

## 4 DEMO Scenario

We create a virtual Karaoke device which is composed from four real UPnP devices: TV, player, speaker, and microphone. Imagine your intelligent home environment has four UPnP devices: TV, speaker, player, and microphone. Each device supports independent services, such as TV has Visual service to display a TV program or a video file from player. Player device has PlayFile service to play any audio and video file from compact disc. Microphone can convert sound into an electrical signal, and usually fed into an amplifier, a recorder, or a speaker. And speaker can play an audio stream. General user without professional knowledge does not know what complex applications can be created from these four UPnP devices. In our system, VAP could create a virtual Karaoke device from these four UPnP devices but you don't really have a real Karaoke device. Fig. 6 shows the creation of virtual Karaoke device from the four devices using interface matching.

### 4.1 Semantic Service Description

At first, we make some description files of devices and services to create UPnP device. The URLs in the device description are relative locations from the directory of the device description file. A service must be able to represent not only data type, input/output but also the semantic information of a service. We add a semantic tag in XML-based service description file. Fig. 7 shows that the right part is the service description with the semantic tag and the left part is the device description. SCPDURL in the device description records the location of service description. When a UPnP device is plugged into the network, the control point would retrieve the device and service descriptions from the discovery message.

### 4.2 UPnP Devices Simulation

We implement UPnP devices and control point by



Fig. 8: Media player



Fig. 9: Virtual Karaoke device

using CyberLink UPnP package [10]. Fig. 8 shows a media player, which have two services: SetPower and PlayFile. The input data type of PlayFile is Bin.base64 and its semantic is File. The output data type of PlayFile is Bin.base64 and its semantic is Audio and Video.

### 4.3 Virtual Karaoke Device Creation

If control point finds the TV, player, speaker, and microphone, service composition system would create a virtual Karaoke device which is composed from these devices automatically. A composite path that found by the VAP is: (Bin.base64, File) → (Player Device, PlayFile) → (Bin.base64, Video) → (TV Device, Visual) → (Device\_Status, TV) and (Bin.base64, Voice) → (Microphone Device, Mike) → (Bin.base64, Audio) → (Speaker Device, PlaySound) → (Device\_Status, Speaker). Fig. 9 shows that users could view and invoke the virtual Karaoke device through control point. When we invoke the virtual Karaoke service, control point would invoke the required services on the execution path automatically.

## 5 Conclusions

In this paper, we present how to use semantic tag to aid dynamic service composition of home UPnP services. We design the service interface with data type and semantic information. Data type and semantic ontology is easy to design service interface for developers and is easy to understand for users. Service interfaces are public for an

external use. A service can hold two sorts of interfaces: input interface and output interface. Each interface is specified by data type tag and semantic tag. With interface matching method, we could know which the service's output can be fed into the next service's input. We also present service graph and VAP to find composite execution paths of virtual device. Service graph is an intuitive way to represent composition concepts in an understandable way. Once service graph is constructed, we use VAP to find a shortest composite execution path in the directed service graph. A shortest execution path means that we use the least services to compose a virtual device. Our dynamic service composition system could create virtual device from the primitive devices in home environment. And applications are no longer restricted to designer's imagination.

## Acknowledgement

This study is conducted under the "Applied Information Services Development & Integration project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China.

## Reference

- [1] Microsoft Corp., Universal Plug and Play Device Architecture, v.1.0, Jun. 2000
- [2] D. Marples and P. Kriens, "The Open Services Gateway Initiative: An Introductory Overview," IEEE Communications Magazine, Dec. 2001
- [3] Digital Living Network Alliance, "DLNA Networked Device Interoperability Guidelines," Mar. 2006
- [4] HAVi, "HAVi, the A/V digital network revolution," White Paper, 1999
- [5] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business Process Execution Language for Web Services," Jul. 2001
- [6] K. Fujii and T. Suda, "Dynamic Service Composition Using Semantic Information," Int'l Conference on Service Oriented Computing, Nov. 2004
- [7] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, M.-C. Shan, "Adaptive and dynamic service composition in eFlow," CAiSE, Mar. 2000
- [8] William K. Cheung, Jiming Liu, Kevin H. Tsang, Raymond K. Wong, "Dynamic Resource Selection for Service Composition in the Grid", Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 2004
- [9] Xiaohui Gu, Klara Nahrstedt, "Dynamic QoS-Aware Multimedia Service Configuration in Ubiquitous Computing Environments", Proceedings of the 22<sup>nd</sup> ICDCS'02
- [10] CyberLink for Java, available online <http://www.cybergarage.org/net/upnp/java/>
- [11] UPnP Forum, <http://www.upnp.org/>

# Using Service-Oriented Architectures for Socio-Cultural Analysis

David Garlan, Kathleen M. Carley, Bradley Schmerl, Michael Bigrigg, and Orieta Celiku

School of Computer Science  
Carnegie Mellon University  
5000 Forbes Ave, Pittsburgh PA 15213 USA  
+1 412 268 5056

{garlan, carley, schmerl, bigrigg, orietac}@cs.cmu.edu

## ABSTRACT

An emergent domain that raises some unique engineering challenges is that of software architectures to convert large bodies of unstructured data to human-usable knowledge, such as in the domain of socio-cultural information analysis. We report on an architecture based on Service-Oriented Architectures that we are applying in this domain. We list the requirements that such an architecture must support, describe our architecture for addressing them, and outline what we believe are the important engineering and research issues that must still be overcome.

## 1. INTRODUCTION

One of the most striking features of today's computing landscape is the exponentially increasing volume of information that is becoming electronically accessible. Finding ways to use this information effectively – to access it in its myriad forms and formats, to extract insight and knowledge, and to update those results as information changes – is a significant software engineering challenge. While systems such as search engines provide important capabilities for accessing and organizing some of this information, there remains a large gap between the huge corpus of largely-unstructured data and human-usable knowledge.

To address this problem a number of researchers are developing a new breed of powerful information analysis tools, called Dynamic Network Analysis, that include capabilities to do natural language processing on large volumes of data, techniques for extracting key relations between entities, and mechanisms for analyzing, filtering, forecasting and visualizing this information as an ecology of evolving networks including social, knowledge and activity networks [1][2][5]. For example, as detailed later, such tools can be used by scientists to understand change in the Sudan, military or intelligence agencies to understand how to interact with allies, organizational analysts to examine changing connections among firms and products as evinced by news stories.

Unfortunately, as implemented today, such tools have a number of severe limitations.

**Stovepiped systems:** Current information analysis systems are often large, monolithic programs that make it difficult to compose their constituent capabilities with those of other systems.

**Restricted models:** Current systems can only work with a limited set of information models that make interchange and coordination problematic.

**Lack of configurability:** Current systems are often tuned to a specific class of analyses or information abstraction, and can only be tailored by users having detailed low-level knowledge of their parameters of operation.

**Idiosyncratic interfaces:** Each system adopts its own interface conventions, requiring users to learn different interaction conventions for each tool.

**Platform restrictions:** Current systems often make rigid assumptions about the specific platform that they can work on, making it difficult to use them in a distributed setting, or to balance the need for secure co-location with access to external capabilities.

**Duplicated functionality:** Current systems are often engineered to work in stand-alone fashion, requiring each system to duplicate functionality also required by others – for example, in support of graphical interfaces, data management, communication, security, etc.

In this paper we describe an approach that addresses these problems. The key idea is the use of a common integration architecture, based on service-oriented architectures, that handles the special requirements for flexible information analysis. Critical to the success of this approach is the strong involvement of the community of tool developers and tool users in identifying standard models and ontologies to support interoperability, within the service-oriented context. Focusing specifically on the domain of socio-cultural analysis, in the remainder of this paper we list those special requirements, describe our architecture for

addressing them, and outline what are the important engineering and research issues that must still be overcome.

## 2. SOCIO-CULTURAL ANALYSIS

Socio-cultural analysis involves understanding, analyzing and predicting the relationships in large complex social systems. Complex social systems are typically represented as dynamic networks that relate entities in the system (e.g., people, knowledge, actions) to each other. The emergent field of dynamic network analysis (DNA) is centered on the collection, analysis, understanding and prediction of dynamic relations in and among networks, and the impact of such dynamics on individual and group behavior. DNA facilitates reasoning about real groups as complex dynamic systems that evolve over time. Within this field computational techniques, such as machine learning and artificial intelligence, are combined with traditional graph and social network theory, and empirical research on human behavior, groups, organizations, and societies to develop and test tools and theories of relational enabled and constrained action.

The application of DNA techniques to a large complex social system, such as the US Army or gang networks, entails a series of procedures. First, one needs to gather the relational data. One approach for doing this is to extract relations from a corpus of texts such as public domain items like web pages, news articles, journal papers, stock holder reports, community rosters, and various forms of human and signals intelligence. Second, the extracted networks need to be analyzed. That is, given the relational data, identifying key actors and sub-groups, points of vulnerability, and so on. Third, given a set of vulnerabilities, we want to ask what would happen to the system were the vulnerabilities to be exploited. How might the networks change with and without strategic intervention?

The center for Computational Analysis of Social and Organizational Systems (CASOS) at Carnegie Mellon University has been engaged in developing methods and tools to achieve these activities. The tools are interoperable and can be organized as a chain to extract networks from texts, ana-

lyze these networks, and then engage in what-if reasoning.

This tool suite takes into account multi-mode, multi-link, and multi-time period data including attributes of nodes and edges. This toolset contains the following tools: AutoMap [4] for extracting networks from natural language texts, ORA for analyzing the extracted networks [3], and Construct for what-if reasoning about the networks.

Figure 1 provides an example of the way that these tools are integrated into a tool chain. Each of the tools (Automap, ORA, Construct) are monolithic programs. They are loosely integrated through an XML format called DyNetML, which is an interchange format for rich social network data.

While the existing tools are powerful, their interaction in terms of a tool chain is coarse-grained because the applications themselves are monolithic. Thus, expert knowledge is required to use each tool. The information shared amongst them in terms of traceability or reproducibility is impoverished, meaning that conducting analysis when new information becomes available, or on entirely new but related datasets, is difficult. Additionally, linking tools developed by other members of the DNA community is challenging.

## 3. ARCHITECTURAL DRIVERS

To overcome the limitations outlined above, we require a platform and architecture within which socio-cultural analysis tools can be integrated, configured, extended and programmed by end users, and tailored to specific domains without extensive low-level expertise.

Specifically, data collection, analysis and modeling tools must reside within an architecture that supports six key requirements [9].

**Heterogeneity** in data sources, analytical models, analysis mechanisms, and end-user needs. As the use of these systems expands, we can assume increasingly diverse sets of elements that will need to be integrated into future systems.

**Flexible configuration** to (a) assemble existing components (data sources, data coding tools, analysis tools, visualization tools, and simulation models) in new ways depending on the type of data available and the kind of analysis needed, (b) add components to support new capabilities, and (c) allow users to easily experiment with new analysis paths, workflows, and simulations without detailed technical knowledge of the tools and underlying technologies.

**High performance** processing and manipulation of large, diverse, and distributed sources of data to allow interactive exploration and analysis.

**Traceability** of analytic output to sources and intermediate models and records in order of processing, to allow analysts to compare results of analysis to ground and derived truth, and to adjust the fidelity and parameters of their models.

**Security and privacy** of potentially sensitive information that is used in the analyses.

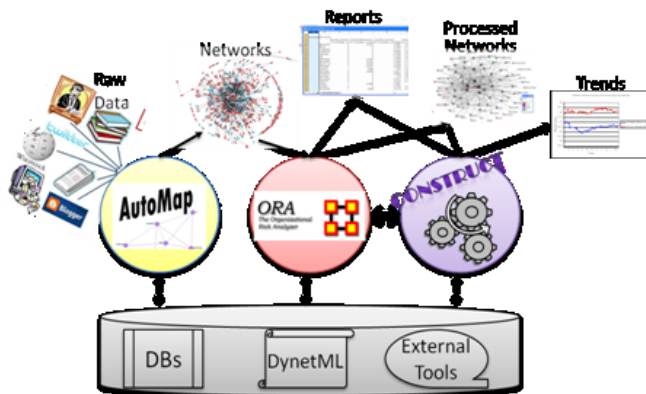


Figure 1. The Toolchain for socio-cultural analysis developed by the CASOS group at Carnegie Mellon.

**What-if reasoning** by enabling the analyst to change how data is coded, what data is coded, what virtual experiments in the simulations are run, track the impact of those decisions, set up multiple choice paths to run in parallel to facilitate rapid assessment making use of data-farming techniques, and replay facilities for desired procedures so that future data sets can be analyzed.

#### 4. RELATED WORK

A number of modern integration frameworks make services accessible. One of these is Web Services technologies that provide standards for interaction, including SOAP [8] and REST [6]. Although standards for Internet-base invocation are a first step towards service integration web service infrastructure does not support ways to define workflows of services, web service lifecycle issues, or dynamically locating of services – capabilities necessary for our domain..

Service-oriented architecture (SOA) [10][11] aims to address some of these issues by defining standards for workflows (called *orchestrations*), policies for governance, and facilities for service discovery. Many definitions and implementations of SOAs aim to be applicable for general business domains. While SOAs provide important capabilities for service coordination, by themselves they have limitations that must be overcome to be applicable to our domain: (a) orchestration scripts define low level coordination, and are not appropriate for use by non-technical users; (b) support for agile and dynamic workflows is often impoverished in existing technologies; and (c) existing technologies have performance-related issues that make them difficult to use in context (such as ours) where large flows of data must be efficiently processed.

As an example of these limitations, consider the standard methods for defining SOA workflows: the Business Process Execution Language (BPEL) [12] and Business Process Modeling Notation (BPMN) [15]. BPEL and BPMN are graphical programming languages that allow specification of general business processes. BPEL especially is intended to be interpreted and therefore requires the detail of a programming language and the skill of a programmer. To address this, the SOA community has introduced a more abstract notation for defining orchestrations called BPMN. The goal is that orchestrations defined in BPMN can be understood by all business users. However, business analysts are still required to define orchestrations in BPMN, rather than non-technical users.

Taking these limitations into consideration, it is necessary to augment SOA technology and concepts to particular domains. For socio-cultural analysis, this is particularly relevant because it is necessary that services should be ultimately assembled by non-technical field analysts who have expertise in the domain they are trying to analyze, but little expertise in programming. Thus, one of the challenges is identifying the abstractions and protocols that should be built on top of SOAs, but that are tailored to the needs of

the socio-engineering analysis domain. Furthermore, we require an easy-to-user approach for service assembly.

Among the other technologies that attempt to provide general-to-use workflow definition in other domains are Yahoo! Pipes [18] for defining mashups on the Internet and uDesign for defining activities in pervasive computing environments [17]. Our work is similar in spirit to these efforts, but specialized for socio-cultural analysts.

There are also several implementations of infrastructures that provide an extensible framework for socio-cultural analysis, particularly in the intelligence analysis domain. For example, COMPOEX [7] provides an integration architecture for assisting military commanders and civilian leaders in selecting models and analyses to plan and execute military campaigns. The goal of our approach is to develop a framework that is targeted more generally at socio-cultural analysis (not limited to military and intelligence activities). Furthermore, COMPOEX is focused on simulation once models have been developed, whereas our approach also includes the ingestion of raw data to produce the models.

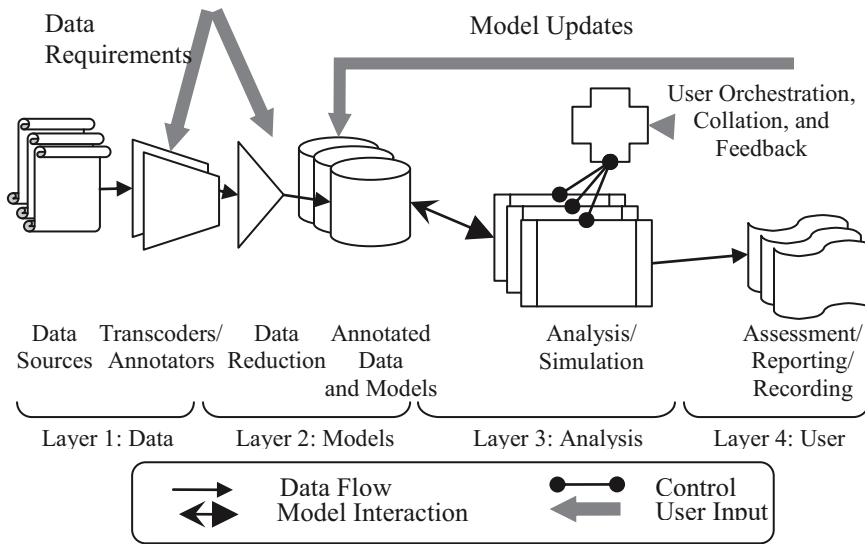
#### 5. ARCHITECTURE DESCRIPTION

From a functional perspective, information analysis systems have a common flow of processing: Data is input into the system originating from many sources. These sources, including public news reports and intelligence reports, are typically written in natural languages. These inputs need to be processed and marked up to identify key concepts that are needed for intelligence analysis, and output in a form that is suitable for simulation and analysis. The concepts, or *entities*, of interest in the inputs are mostly fixed for the domain, and include knowledge and agents. The output from processing is a model represented as a graph with relationships among agents and knowledge. Models can then be analyzed and viewed in a variety of ways. Further, simulation or what-if analysis may be performed to create a set of related models. Insight gained is then used to refine the data processing and analysis. Finally, reports of various kinds can be generated and stored.

Our architecture naturally follows this decomposition of activities, while building on best practices in the engineering of service-oriented architectures and new techniques to support end-user programming and system configuration. The basis for the architectures is (a) the use of a multi-layered system capturing the essential flows of information and processing, and (b) support for flexible orchestration, coordination, and transformation.

**Multi-layered system.** The domain of dynamic network analysis naturally lends itself to a four-tiered system shown in Figure 2:

1. *Data Layer:* a set of heterogeneous data sources. These include databases, wire feeds, intelligence streams, email corpuses, web sites, historical documents, etc. These form the raw inputs to the system, and may be relatively stable



**Figure 2. High Level View of SORACS Architecture.**

(as in the case of historical databases), changeable (as in the case of web sites), or highly dynamic (as in the case of wire feeds and intelligence streams).

2. *Model Layer*: a set of high-level models, which represent information extracted from the first layer. This layer will be populated by a variety of models including annotated documents (e.g., as a result of natural language processing) and network models (e.g., ORA Meta-Networks 0) representing relationships between key entities in the domain of discourse. Bridging these two layers is a set of model extractors (e.g., Automap [4], CEMap) that effect the transformation of raw data into theoretically richer forms for analysis.

3. *Analysis Layer*: populated by a collection of analysis tools (including ORA, UCINET, Pythia). Such tools will reside as semi-independent components, interacting with models in the second layer and generating input for and analyzing results from tools in the fourth layer through a standard set of protocols. This layer also includes simulations, such as Construct, for forecasting and exploring alternative histories and futures. Simulation and analysis tools will have well-defined interfaces, and be integrated into a service-oriented framework that enables registry and lookup in support of dynamic configuration and incremental reconfiguration.

4. *User Layer*: the end-user layer, which provides an interface for users to interactively view the analysis results, configure new analyses, trace analysis to sources, and generate reports. Capabilities in this layer fall into two categories. One is output of analyses and simulations from the lower layer (such as ORA reports); this also allows the user to fine-tune the parameters of these (e.g., specifying whether reports will be generated for the entire network or key entities). The other supports orchestration, allowing users to put together new combinations of processing that determine

both the nature of model generation and the way in which analysis/simulation services are assembled.

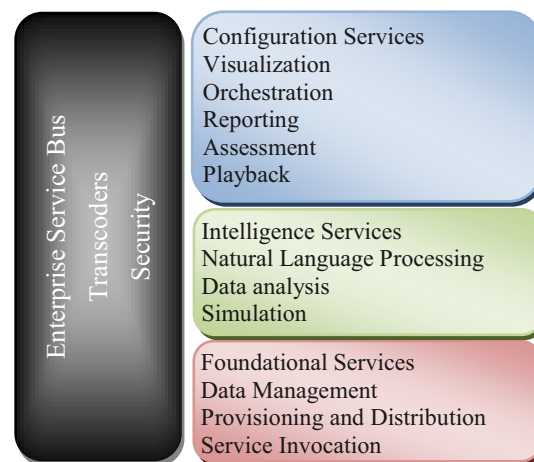
**Orchestration, Coordination, and Transformation.** To enable dynamic integration and configuration of the components into the four layers requires a number of mechanisms for orchestration, coordination and transformation.

First is the ability to automatically produce full analysis pipelines from end-user descriptions that specify at a high level of abstraction what kinds of processing is needed to produce a particular kind of analysis. Using a combination of graphical and textual inputs, users will be able to specify and configure a collection of data transformation and analyses services to support their needs. The system automatically assembles these parts, providing

the “glue” for connecting the parts.

Second is the ability to select the appropriate transcoders to bridge data-mismatch assumptions between components. Building on earlier research in document transformation, the platform will include a registry of transcoders and filters, together with algorithms that find an optimal chain of transformations (based on information fidelity metrics) [13][14]. While manual fine-tuning of these transformations may be necessary in some cases, we expect that the majority can be done automatically.

Third is the use of a standards-based service-oriented architecture for the analysis tools. Specifically, the architecture contains a registry of the services provided by the suite of existing and newly-developed tools. As users compose dataflow paths for analysis, services are automatically selected and composed in appropriate ways (in some cases requiring the automatic interposition of data transformers).



**Figure 3. Layered Service Oriented Architecture.**

In addition the integration framework will provide a set of common services for communication, security, provenance, and mismatch avoidance.

The organization of these services follows a fairly standard approach used by modern SOAs. As illustrated in Figure 3, we organize the services into three groups. At the bottom are foundational services, including data management, provisioning and distribution, and service invocation. Next are services that provide the meat of the processing. This is where tools specific to the data transformation, analysis, and simulation for understanding and interpreting information. At the top are configuration service, which support the specification and tailoring of computations, as well as certain visualization services for presenting information to a user. Communication and coordination is handled by an Enterprise Service Bus, which supports service discovery, look-up, enlistment, and interaction.

## 6. IMPLEMENTATION

The tools described in Section 2 encompass a wide range of activities required by socio-cultural analysts. They are therefore good candidates for the initial investigation of an architecture in this domain. In this section we outline our current implementation of an initial version of the above architecture using the existing tools developed by CASOS. The initial step in our investigation is identifying and implementing the individual services that can be derived from these tools from which orchestrations can be derived.

Automap analyses textual data. It can process data lexically (e.g., by removing extraneous white space, splitting sentences) and grammatically (e.g., by identifying and extracting parts of speech, resolving pronouns). Services derived from Automap can be considered lexical services and grammatical services or simply a combined textual service. Service-able functionality also exists within ORA. ORA contains many different common network science metrics and grouping algorithms (e.g., CONCOR, Newman, FOG, Johnson Hierarchical, Attribute based). It also has facilities for generating, editing, visualizing, and detecting changes

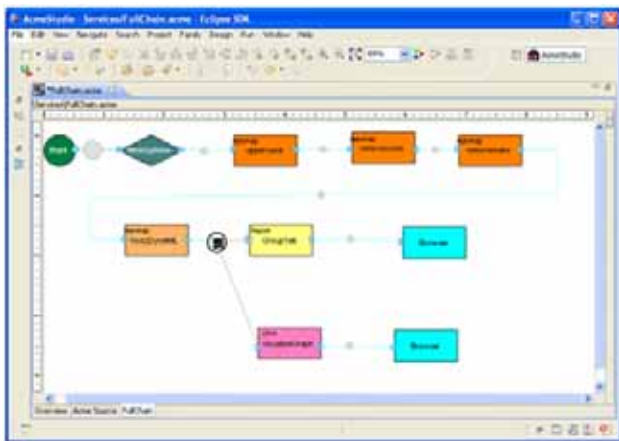


Figure 4. The AcmeStudio orchestration interface.

in networks. Construct includes services such as experimental design construction, report generation and simulation. In all cases, the services can be provided at a fine or coarse grain level in which analysis operations are provided as a graph analysis service.

Once the services were identified, we entered a rearchitecting phase that involved making the services more decoupled in the tools. For Automap, this involved making available each of the lexical analysis components as independent components decoupled from the existing user interface; for ORA it involved decoupling the analyses and reports desired from the user interface. This process is ongoing, and we present a discussion of the challenges in the following section.

Once the functions that could be used as services were identified and isolated in the code, we then implemented them as services using the standard approach to web service definition (WSDLs) and using an open source application server to make these available as web services. For this phase, we used Apache Tomcat as our application server, and Apache CXF to streamline our implementation of the existing Java implementations as web services.

Our initial version of the orchestration interface for this domain has been written as a plug-in to AcmeStudio (see Figure 4) [16]. We have defined an architectural style for this domain, detailing each of the services as particular component types, and defining connector types that are specific to this domain allowing the components to be chained together. The orchestration backend of the plug-in takes architectural specifications and produces BPEL definitions that can be uploaded and executed by a BPEL engine. We currently use Apache ODE as our orchestration engine for executing these orchestrations.

## 7. DISCUSSION & CONCLUSION

In this paper, we have discussed the requirements and design for an architecture for socio-cultural analysis. We have also described an initial implementation that provides a domain-specific approach to defining workflows that is built upon existing SOA standard technologies. In doing this, we encountered a number of additional issues and technological challenges that are imposed by this domain.

*What are the appropriate connectors for long-lived service invocations?* While the intent in SOAs is for service orchestrations to execute over long periods of time (e.g., many years), support for invocation of long running individual services is low. BPEL provides a way to deal with long running services through asynchronous invocation; the BPEL program then polls for results. The domain of socio-cultural analysis must support long running services because the amount of data being analyzed is typically large, and the analyses complex. However, the simplest model for analysts is to define call-return type connections between services. There needs to be a balance between conceptual ease for analysts and technical detail for developers.

Related to this is the issue of *control vs. usability*. How do we define services that have the appropriate interfaces for use in the common case, but still provide enough control for detail-oriented analysts? For many analyses in this domain, there are a set of common or default parameters that are sufficient in most cases, but we still wish to provide control for the less common cases.

*Traceability and reproducibility* is another challenge that we need to address. SOA platforms provide some coarse-grained traceability through provenance mechanisms. However, we require finer grained traceability so that analysts can query how analysis conclusions were made, and the reliability of the data that they were based on. Furthermore, we require the ability to rerun orchestrations with minor changes in data. Currently, there is no mechanism for providing incremental analysis or data-caching to reduce the time these analyses take.

These challenges are areas of future work. Moreover, we are planning to extend the current prototype in a number of directions, including: a) providing automated transcoding between data formats, and b) allowing the definition and reuse of workflow templates.

Another area of future work evaluating the effectiveness of the architecture for the socio-cultural domain. The architecture described in this paper matches the way that analysts think about the problem; we believe that the technical aspects balance the needs of users and developers of tools. We have some confidence that the architecture is correct through integration of existing CASOS tools. Future work will involve integrating additional components, and developing a more functional user interface for analysts to use.

## ACKNOWLEDGEMENTS

This work is supported in part by the Office of Naval Research (ONR), United States Navy, N000140811223 as part of the HSCB project under OSD. Additional support for the core CASOS tools was provided by National Science Foundation (NSF) Integrative Graduate Education and Research Traineeship (IGERT) program, NSF 045 2598, the Air Force Office of Scientific Research, FA9550-05-1-0388 under a MURI on Computational Modeling of Cultural Dimensions in Adversary Organizations, the Army Research Institute W91WAW07C0063, the Army Research Lab DAAD19-01-2-0009, the office of Naval Research (ONR), N00014-06-1-0104 the Army Research Office W911NF-07-1-0060, and CASOS - the center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University (<http://www.casos.cs.cmu.edu>). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, the Office of the Secretary of Defense, the Army Research Lab, the Air Force Office of Sponsored Research, the Army Research Office, the National Science Foundation or the U.S. government.

## REFERENCES

[1] Carley, K.M., Dynamic Network Analysis” Dynamic Social Network Modeling and Analysis: Workshop Summary and Pa-

pers. Breiger, R., Carley, K., Pattison, P. (eds). Committee on Human Factors, National Research Council, 2003.

[2] Carley, K.M., A Dynamic Network Approach to the Assessment of Terrorist Groups and The Impact of Alternative Courses of Action. In Visualizing Network Information Meeting Proceedings RTO-MP-IST-063, Neuilly-sur-Seine, France, 2006.

[3] Carley, K.M., Columbus, D., DeReno, M., Reminga, J., and Moon, I.-C. ORA User’s Guide 2007. Carnegie Mellon University School of Computer Science Institute for Software Research Technical Report CMU-ISR-07-115, 2007.

[4] Carley, K.M., Columbus, D., DeReno, Diesner, J., and Sebul, N. AutoMap User’s Guide 2007. Carnegie Mellon University School of Computer Science Institute for Software Research Technical Report CMU-ISR-07-114, 2007.

[5] Carley, K.M., Diesner, J., Reminga, J., and Tsvetovat, M, Toward an Interoperable Dynamic Network Analysis Toolkit, DSS Special Issue on Cyberinfrastructure for Homeland Security: Advances in Information Sharing, Data Mining, and Collaboration Systems, 43(4), p. 1324 – 1347, 2007.

[6] Fielding, R. Architectural Styles and the Design of Network-based Software Architectures. Ph.D. Thesis, University of California, Irvine, 2000.

[7] Kott, A. and Corpac, P.S. COMPOEX Technology to Assist Leaders in Planning and Executing Campaigns in Complex Operational Environments. In 12<sup>th</sup> International Command and Control Research and Technology Symposium, 2007.

[8] Mitra, N., and Lafon, Y. (eds) SOAP Version 1.2 Specification (Second Edition). <http://www.w3.org/TR.2007/REC-soap12-part0-20070427>, 2007.

[9] Naval Research Laboratory (NRL) Diplomatic, Informational, Military, Economic (DIME) Political, Military, Economic, Social, Information, Infrastructure (PMESII) Modeling Requirements Workshop. John Hopkins University, Maryland, 2007.

[10] Newcomer, E., Lomov G. *Understanding SOA with Web Services*. Addison Wesley, 2005.

[11] OASIS. OASIS Reference Model for Service Oriented Architecture 1.0. <http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tn014.pdf>. 2006.

[12] OASIS. Web Services Business Process Execution Language Version 2.0, April 2007. URL <http://docs.oasis-open.org/wsbpel/2.0/wsbpelv2.0.html>.

[13] Ockerbloom, J. Exploiting Structured Data in Wide-Area Information Systems, Ph.D. Thesis. Carnegie Mellon University Technical Report CMU-CS-95-184, August, 1995

[14] Ockerbloom, J. Accommodation: The Key to Making Widely Adopted Composable Systems. In Proc. Workshop on Compositional Software Architectures, Monterey, CA, 1998.

[15] Object Management Group. The Business Process Modeling Notation (BPMN) Version 1.2. January, 2009. URL: <http://www.omg.org/docs/formal/09-01-03.pdf>.

[16] Schmerl, B., and Garlan, D. AcmeStudio: Supporting Style-Centered Architecture Development (Research Demonstration) In Proc. the 26th ICSE, Edinburgh, Scotland, May 2004.

[17] Sousa, J.P., Schmerl, B., Poladian, V. and Brodsky, A. uDesign: End-User Design Applied to Monitoring and Control Applications for Smart Spaces. In Proceedings of the 2008 Working IFIP/IEEE Conference on Software Architecture, Vancouver, BC, Canada, 18-22 February 2008

[18] Yahoo!, Inc. Yahoo Pipes. <http://pipes.yahoo.com/pipes>. Accessed March, 2009.



# A Conceptual Model for Comprehension of Object-Oriented Interactive Systems

Izuru Kume  
Nara Institute of Science and Technology  
Graduate School of Information Science  
8916-5 Takayama, Ikoma,  
Nara 630-0192, Japan  
kume@is.naist.jp

Etsuya Shibayama  
The University of Tokyo  
Information Technology Center  
2-11-16 Yayoi, Bunkyo,  
Tokyo 113-8658, Japan  
etsuya@ecc.u-tokyo.ac.jp

## Abstract

*Event-driven programming often takes advantage of effects to implement interactive features. Understanding the role of objects from the viewpoint of effects is important to understand the implementation of interactive features. However, currently no existing methods support comprehension of object's roles in taking advantage of effects.*

*In this paper, we propose a modeling framework that becomes the basis of a trace analysis method to comprehend the roles of objects in taking advantage of effects in a feature implementation. The proposed framework provides maintainers a low-level trace model to represent runtime effects 'as is', and a high-level model to abstract the effects according to maintainers' concerns. The framework also defines analysis commands to construct such abstraction of runtime effects. Using the analysis commands, we describe a practical analysis scenario to detect a method execution that triggers the calculation of an output of a feature using previous user inputs.*

## 1. Introduction

Object-oriented systems, such as CASE tools manipulated by GUIs and Java servlet applications that respond to clients' requests, usually adopt an event-driven programming style to implement their *features* [1] with multiple user inputs. In an execution of an event-driven program, event handlers, which were registered in a framework such as Java Swing in advance, are invoked from the framework and process a given input. The event handlers rarely invoke each other, and thus their execution is mutually independent from the viewpoint of control-flows.

An *effect* is a change of runtime state caused by assigning a new value into a *persistent variable*, which is either an instance variable, a class variable or an array component. Programmers often take advantage of effects to make

an input handling process started by an event handler under influence of previous input handling processes. Such a use of effects are necessary to implement a feature that requires multiple inputs to produce an output.

For example, the drawing feature of a drawing program requires two user inputs, one for selection of a drawn figure and another for indication of a drawing place. When we examine the execution of the drawing program, we can find that the figure selection is assigned to an instance variable, which is then referenced when the point indication is processed. It is a simple example of *coupling by effects*, which enables a kind of data exchange among event handlers. Our final research goal is to establish a comprehension method to support understanding of couplings by effect of features of object-oriented interactive systems.

Understanding the role of objects that participate in an object collaboration is important for program understanding [10], and several trace analysis methods (e.g. [9]) are proposed to support the understanding of such roles. Well-accepted basic object-oriented modeling concepts such as collaboration and responsibility [11] become the background of such supporting methods. On the other hand, as for the understanding of coupling by effects, we have no theoretical background in spite of the importance to comprehend effects in the implementation of interactive features, which makes effects comprehension difficult.

To solve the problem, we propose a modeling framework that supports abstraction of a coupling by effects, and the roles of the participants of the coupling by effects. Our framework provides: (1) a model of low-level execution traces on which couplings by effects are represented 'as is', (2) an abstraction framework to comprehend objects' roles in a coupling by effects, and (3) analysis commands to construct high-level abstraction to comprehend objects' roles. We also introduce a practical analysis scenario which is useful to detect objects that plays the role of 'trigger' of processing multiple inputs, and is applicable to various cases. We demonstrate the expressive power of our proposed ab-

straction framework by describing the practical scenario by our framework.

## 2. Preliminary

*Production of an effect* means an assignment into some persistent variable. *Use of an effect* means a reference of the persistent variable whose assignment causes the effect. For a persistent variable holder  $H$ , an *effect on  $H$*  means the effect caused by one of  $H$ 's persistent variable.

An *operation* is a unit of indivisible execution such as arithmetic operations. An *event* is a pair of an operation and its result. Events are the primitive elements to represent a program execution. An *action* is an execution of a sequence of one or more program statements. Events are so fine grained that explanation in terms of events is not intuitive. Therefore, we often explain our modeling concepts in terms of actions and method executions.

For a method execution  $M$ , the *method process under  $M$*  is a sequence of method executions triggered by  $M$ . When  $M$  is an execution of an event handler for an input  $I$ , we call the method series under  $M$  the *handling process of  $I$* . For sum condition  $C$  on method executions, we say that a method invocation  $M$  is a *topmost method execution that satisfies  $C$*  if  $M$  is included in a method process under  $M'$  which is not  $M$  and satisfies  $C$ .

## 3. Motivating Example

Throughout this paper, we use an existing object-oriented interactive system that is built on Java Swing framework, and that provides an interactive feature to accomplish an one-on-one battle between two RPG (Role Playing Game) characters. Readers can find more detailed explanation in [3].

When the program is started, two panels each of which represents an RPG character are displayed. Each panel displays the status of the represented character (the character's name, HP value, and etc.), and a button to send a command to attack the opponent. Clicking only one button produces no output, but successively clicking the another button triggers an one-on-one battle which includes success-failure checks of attacks and damage calculation, and results in an update on the status of the both characters.

The sequence diagram in Figure 1 focuses on the interactions among the main class for initialization, a battle command receptor `battle_field`, and two attack command objects (`command_1` and `command_2`) in the whole process to accomplish an one-on-one battle. Actions that accompany the process are indicated in the balloons in Figure 1.

The execution starts with the system initialization by the main class. In the process of the initialization, the com-

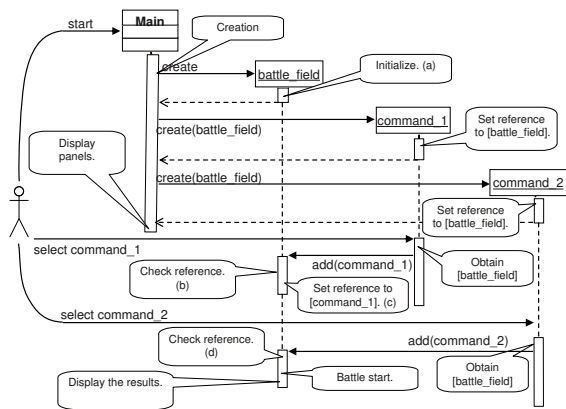


Figure 1. Sequence Diagram with Actions

mand receptor (`battle_field`) is created and initialized. All instance variables of the receptor are initialized at this time. (a) Then two attack commands (`command_1` and `command_2`) are created and initialized. At their initialization, the command receptor is assigned to their instance variables. Then two panels are displayed and the system initialization finishes.

On clicking the button on one panel, a method is invoked on `command_1` which then invokes a method `add` of the command receptor (`battle_field`) with itself as one of the method argument. The command receptor receives an attack command through the method `add`. During the execution of `add`, the command receptor performs an conditional branch based on its state (b).

If the state had been changed since the initialization, then it would obtain the previously received battle command by referencing the instance variable, and would start the one-on-one battle which uses the obtained battle command together with the just received battle command by the method invocation. At this time, the state remains unchanged since the initialization, and it assigns the received attack command (`command_1`) to one of its instance variable (c).

On the successive clicking on the another button, another attack command `command_2` similarly invokes the method `add` on the command receptor. When the command receptor performs the conditional branch again (d), its state is updated at the assignment (c). Thus the command receptor starts the one-on-one battle and completes the second input process.

From the above explanation, we can easily see that the first conditional branch (b) is influenced by the state initialization (b), and the second conditional branch (d) is influenced by the instance variable assignment (c). It means that the command receptor participates in the coupling by effects of the one-on-one battle. Similarly the two attack commands `command_1` and `command_2` also participate in the coupling by effects, because at the input han-

dling processes they use their effects at the system initialization. There are many other participants in the coupling by effects although they are not depicted in the sequence diagram.

Let's assume a situation that a maintainer tries to locate the code that calculates the success-failure checks of the attacks and damaged based on the two attack commands. The maintainer only knows that the system starts the calculation process as a result of one or more conditional branches somewhere in the source code that checks an attack command has been sent or not at every input handling process. The maintainer has to identify the object(s) with the role of calculation trigger in the coupling by effects, which **we know** is the command receptor `battle_field` in the second input handling process.

The maintainer can examine by going around many method executions by applying the omniscient debugger [4] which tells the execution point where an instance variable is assigned the current value, for example. However, such going around means a waste of time and efforts for the maintainer who lacks the knowledge to decide the meaning of the assignment and the reference of the instance variable at the first glance.

The maintainer really requires a method that efficiently identifies the attack command receptor as the trigger of the one-on-one battle calculations by an ambiguous style of conditions such as "the object that is the method receiver of the topmost method execution influenced by the first input handling process", instead of a thorough investigation of numerous instance variables and conditional branches. Now it's a time for us to state our analysis target and our research challenge.

**Analysis Target:** A feature with multiple user inputs has a topmost method execution (1) that produces an output of the feature, and (2) that contains an action influenced by previous user inputs. We call such a method execution *an output trigger*.

**Research challenge:** We pursue a systematic analysis method that enables maintainers to identify output triggers with as little as prerequisite knowledge and efforts of going and backing of method executions.

As for our solution to our research challenge, we show an analysis scenario in section 5 and a conceptual model to formally define the scenario in section 4.

## 4. Analysis Concepts

In this section, we introduce (1) a low-level trace model to represent a coupling by effects at runtime 'as is', (2) an abstraction concepts to understand the role of objects that participate in a coupling by effects, and (3) analysis commands to accomplish the abstraction of a coupling by effects. As for the trace model, readers can find a detailed

explanation in [3] and another application for defects locating problem in [2] for a special case of so called feature interactions.

### 4.1. Trace Model

The trace model specifies the data elements and their relationships in the trace of a program execution. A trace records a sequence of method executions. Each method execution is divided to basic block executions, each of which is further divided to a sequence of events. Method executions, basic block executions, and events (and their operations and results) are represented as independent elements in a trace, which are called *trace elements*. Trace elements have their own attributes and reference each other.

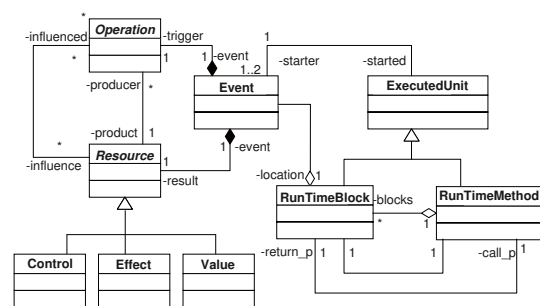


Figure 2. Trace Elements

The class diagram in Figure 2 represents the types of trace elements and their mutual reference relationships. Class `RunTimeMethod` and class `RunTimeBlock` represent method executions and basic block executions respectively. `Event`, `Operation`, and `Resource` represent events, operations, and operation results respectively. Operation results are categorized into (1) value creation (Notice that the result is not the value itself but its creation.), (2) change of control, and (3) effect production. Classes `Value`, `Control`, and `Effect` corresponds to the classification of the operation results.

Operations are categorized according to their resulting types. Value creating operations include constant values, arithmetic operations, class instance creations, array creations, value copies (an abstraction of local variable assignments), and persistent variable references. Control changing operations include method invocations, returns from invocation, and conditional branches. Effect producing operations include persistent variable assignments.

A method invocation event refers to the corresponding method execution. A return event refers to the basic block execution from which the execution continues. Such reference relationships are represented by a `starter-started` association among

Event and ExecutedUnit that abstracts method executions and the basic block execution. Combination of the starter-started association with an event-location association between Event and RunTimeBlock to show which basic block an event belongs to, and the decomposition relationship among RunTimeMethod and RunTimeBlock enables bidirectional navigation between events and method executions.

The trace model defines influences among operations via their results. For example, the effect of a persistent variable assignment influences references of the variable because the assignment decides the value of the variable. A conditional branch clearly influences the operations contained in the basic blocks selected by the conditional branch. We can derive influences among events from the influences of their operations and results. The derived influences constitutes a coupling by effect in a trace as explained below.

When an event  $e_s$  influences another event  $e_d$ , we call the influence-influenced relationship between  $e_s$  and  $e_d$  an *influence direction from  $e_s$  to  $e_d$* , and express it by  $e \Rightarrow e'$ . We call a transitive closure of influence directions from  $e_s$  to  $e_d$  an *influence flow from  $e_s$  to  $e_d$* , and express it by  $e_s \Rightarrow^* e_d$ . Let's  $\{M_i\}_i$  be a collection of method executions in multiple input handling processes. We say that  $\{M_i\}_i$  is *coupled* if the events in  $\cup_i M_i$  forms a connected directed graph with their influence directions. Intuitively speaking, the above condition states that any method execution  $M_i$  is influenced by some of its predecessors (if any).

The union of all existing influence flows among the events in  $\cup_i M_i$  represents the coupling by effects of the  $\{M_i\}_i$  'as is'.

## 4.2. Comprehension Model

As we saw in section 4.1, a coupling by effects primarily consists of events and their influence flows. Objects indirectly participate in the coupling through a connection to participating events. For example, an object connects to an instance variable assignment as the variable holder, or connect to another event that is executed in the execution of the object's method. Therefore, the roles of objects depend to and can be derived from the connection to participating events. In the rest of the paper we concentrate on participating events.

In the following we introduce an abstraction of coupling by effects. The basic idea of our abstraction is very simple: *Designate* a small number of events that has an important role in the coupling by effects, and abstract out the rests by a small number of event sets, called *event chunks*. We regard actions and method executions event chunks represented by their contained events.

Given a set of designated events  $\mathcal{E}$  and a set of event chunks  $\mathcal{C}$ , we can derive influence directions among  $\mathcal{E} \cup \mathcal{C}$

from the influence directions among events. First, select a set of influence directions  $S$  that connects the designated events and anonymous events in the event chunks:

$$S = \{e \Rightarrow e' \mid e, e' \in \mathcal{E} \cup (\cup \mathcal{C})\}$$

Then we replace each  $e \Rightarrow e'$  in  $S$  with a set of derived influence directions  $\text{iflow}(\mathcal{E}, \mathcal{C})$  as follows:

$$\text{iflow}(\mathcal{E}, \mathcal{C}) = \{\text{ext}(e) \Rightarrow \text{ext}(e') \mid e \Rightarrow e' \in S\}$$

where  $\text{ext}(e)$  is defined as follows:

$$\text{ext}(e) = \begin{cases} \{e\} & \text{if } e \in \mathcal{E}, \\ \min(e) & \text{otherwise.} \end{cases}$$

where  $\min(e)$  is the set of smallest event chunks that include  $e$  and is formally defined as follows:

$$\min(e) = \{C \in \mathcal{C} \mid e \in C, \nexists C' \in \mathcal{C}[e \in C' \text{ and } C' \subset C]\}$$

Now we can define *the abstraction of a coupling by effects* by designated events  $\mathcal{E}$  and event chunks  $\mathcal{C}$  as the union of all transitive closures of the derived influence directions in  $\text{iflow}(\mathcal{E}, \mathcal{C})$ .

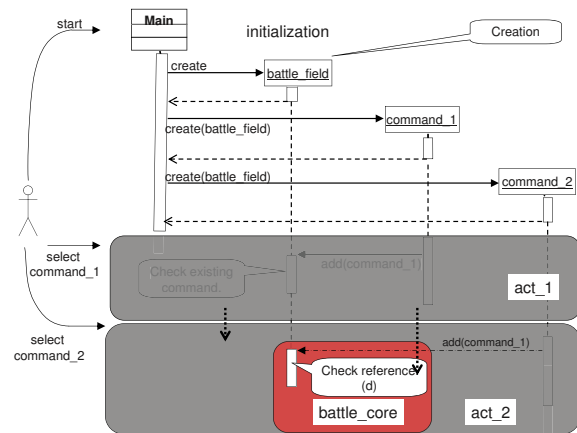


Figure 3. Influences among Event Chunks

Figure 3 shows the abstraction of coupling by effects by the set of designated events  $\{(d)\}$  and the event chunks  $\{\text{act}_1, \text{act}_2, \text{battle\_core}\}$ . Event chunks  $\text{act}_1$  and  $\text{act}_2$  abstract out the events in the first input handling process and in the second input handling process respectively. The only designated event ( $e$ ) is the very conditional branch that decides to start the one-on-one battle calculations. The event chunk  $\text{battle\_core}$  is the only output trigger in the one-on-one battle feature, our analysis target described in section 3.

Abstracting a coupling by effects in terms of actions enables more detailed comprehension of effects and their influences. Figure 4 shows influence flows among the actions in Figure 1. The dashed arrows represent influence

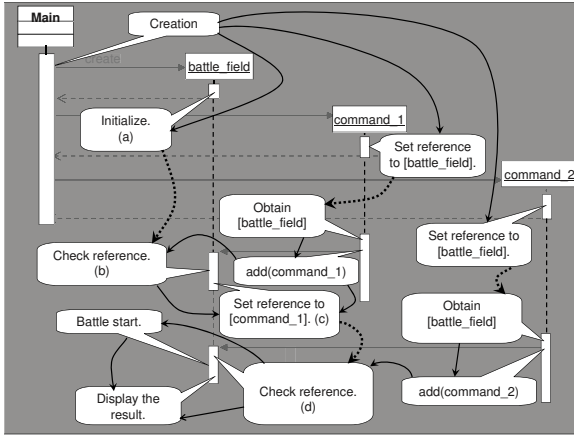


Figure 4. Influences among Actions

flows among different input handling processes. We pursue a means of abstracting a coupling by effects in terms of actions in future.

### 4.3. Analysis Commands

In this section, we introduce analysis commands to introduce a new event chunk or to derive designated events and event chunks from existing event chunks.

**Introduction of event chunks:** Our abstraction framework does not define how to introduce a new event chunk. A means of introducing event chunks should be provided by a trace analysis tool, and thus is implementation dependent. It is desirable that maintainers can introduce event chunks in terms of program elements such as “a set of assignments to the instance variable `age` declared in class `Person`”. In [3] we propose to take advantage of a trace querying language to introduce event chunks.

**Designation:** Designating events is also an implementation dependent matter. It is desirable that a trace analysis tool should provide a way to designate events interactively so that maintainers can check each event contained in a small event chunks. From our experience, designating not only events but also method executions is useful. Therefore, we extend the designation target to method executions. In [3] we propose a prototype analysis tool that provides GUIs to manipulate a small event chunk (or a small method execution set) and contained events (or contained method executions).

**Coupling extraction:** Given a two event chunks  $\mathcal{E}_1$  and  $\mathcal{E}_2$ , the coupling extraction command derives an event chunk  $\text{coupled}(\mathcal{E}_1, \mathcal{E}_2)$  defined as follows:

$$\text{coupled}(\mathcal{E}_1, \mathcal{E}_2) = \{e \in \mathcal{E}_1 \mid \exists e' \in \mathcal{E}_2 \text{ s.t. } e' \xrightarrow{*} e\}$$

**Coverage:** Given an event chunk  $\mathcal{E}$ , the coverage command derives the least set of method executions  $\text{cover}(\mathcal{E})$  that covers all events in  $\mathcal{E}$ :

$$\text{cover}(\mathcal{E}) = \{m \mid \exists e \in \mathcal{E} \text{ s.t. } m \text{ executes } e.\}$$

**Topmost Set:** Given a set of method executions  $\mathcal{M}$ , the topmost command derives a set of the topmost method executions  $\text{top}(\mathcal{M}) \subseteq \mathcal{M}$ :

$$\text{top}(\mathcal{M}) = \{m \in \mathcal{M} \mid \neg \exists m' \in \mathcal{M} \text{ s.t. } \text{under}(m', m)\}$$

where  $\text{under}(m', m)$  means  $m \neq m'$  and  $m$  is contained in the method process under  $m'$ .

**Image:** Given a set of method executions  $M$ , the image command derives an event chunk  $\text{image}(M)$  that are covered by  $M$ :

$$\text{image}(M) = \{e : \text{Event} \mid \exists m \text{ s.t. } \text{under}(M, m) \text{ and } e \in m\}$$

## 5. ‘Output Trigger Detection’ Scenario

In this section we define an analysis scenario that systematically identify output triggers of features in terms of the analysis commands in section 4.3. It is our solution to our research challenge stated in section 3. The analysis scenario, called the ‘output trigger detection’ scenario, aims at locating the topmost method executions that are influenced by previous input handling processes. For the ease of explanation, we pose a restriction that the analyzed interactive feature takes two events, although we can easily extend the scenario for the cases of more than two inputs.

Shortly to say, the ‘trigger detection’ scenario derives the topmost method executions that are in the second input handling process and that are influenced by the first input handling process. The derivation is accomplished by the four steps explained below. For each of the four steps, we explain the result obtained by applying the step to our one-on-one battle example, and how to implement the step in terms of our analysis commands.

**(1) Introduce two event chunks that abstract out the two input handling processes.** By applying this step to our example, we obtain two event chunks `act_1` and `act_2` in figure 3. To accomplish this step, we first introduce a set of executions of event handlers<sup>1</sup>, and designate each execution of the event handlers. Next, we apply the image command to each of the designated elements to obtain two event chunks.

**(2) Extract the event chunk from the second input handling process that are influenced by the first input handling process.** By applying this step to our example, we obtain an event chunk that contains the conditional branch

<sup>1</sup>Remember that we regard a set of method invocations as an event chunk.

(d) in Figure 3. This step is accomplished by a simple application of coupling extraction command.

**(3) Obtain the method executions where the events influenced by the first input handling process are executed.** By applying this step to our example, we obtain a subset of the method process under the second execution of `add` on the command receptor (`battle.field`). This step is accomplished simply by applying the cover command on the result of step (2).

**(4) Obtain the topmost method executions that are in the second input handling process and that are influenced by the first input handling process.** By applying this step to our example, we obtain the set of method executions whose only element is the second executions of method `add` on the commands receptor (`battle.field`). You can optionally designate the conditional branch (d) from the image of this method execution. This step is accomplished simply by applying the topmost command to the result of step (3).

Notice that the steps from (2) to (4) requires no class names nor no method names. According to the implementation by our prototype analysis tool[3], only step (1) requires a knowledge about Java Swing API. As a result, the only programming knowledge required for maintainers to apply the ‘trigger detection’ scenario is the API of the framework on which the system is built on. It is worth noticing that the little requirement on prerequisite knowledge shows a great practical merit of the scenario. For more details of the implementation of the ‘trigger detection’ scenario, see [3].

## 6. Related Work

As for existing analysis methods about effects, object flow analysis [5] shows a great success on test case management [6] and feature dependency analysis [8]. However, because of the difference in th focuses on application areas, the object flow model does not address conditional branches [7], and does not provide an abstraction higher than object aliases. Therefore, it is difficult to obtain a high level representation like ours for to construct a trace analysis method based on the object flow model,

## 7. Conclusion

In this paper, we discussed the problem to comprehend the use of effects and their influences to implement features in a event-driven programming style. As our comprehension target, we defined *output triggers* that start a calculation process to produce a feature output from previously given user inputs. We introduced a practical analysis scenario to systematically detect output triggers. As for the theoretical basis of our approach, we introduced (1) a model of

program execution traces that gives a low-level representation of effects and their influences, (2) an abstraction model to comprehend a low-level representation, and (3) analysis commands to achieve an abstraction. We described the scenario to detect output triggers by the analysis commands.

## Acknowledgement

This research was partially supported by the Ministry of Education, Culture, Sports, Science, and Technology, Grant-in-Aid for Scientific Research on Priority Areas 18049027. We are deeply grateful for the great support and encouragement by Toshiyuki Amagasa, Masahide Nakamura, Jun Miyazaki, Kenji Hatano, Naoya Nitta, professor Yasuhiro Takemura and professor Koji Torii.

## References

- [1] T. Eisenbarth, R. Koschke, and D. Simon. Locating features in source code. *IEEE Transactions on Software Engineering*, 29(3):210–224, March 2003.
- [2] I. Kume and E. Shibayama. Feature interactions in object-oriented effect systems from a viewpoint of program comprehension. In *International Conference on Feature Interactions*, 2009.
- [3] I. Kume and E. Shibayama. A trace analysis scenario to comprehend effects in object-oriented interactive systems. Technical Report NAIST-IS-TR2009003, Nara Institute of Science and Technology, 2009. <http://isw3.naist.jp/IS/TechReport/>.
- [4] B. Lewis. Debugging backwards in time. In *International Workshop on Automated Debugging (AADEBUG)*, 2003.
- [5] A. Lienhard, S. Ducasse, T. Gırba, and O. Nierstrasz. Capturing how objects flow at runtime. In *International Workshop on Program Comprehension through Dynamic Analysis (PCODA)*, pages 39–43, 2006.
- [6] A. Lienhard, T. Gırba, O. Greevy, and O. Nierstrasz. Exposing side effects in execution traces. In *International Workshop on Program Comprehension through Dynamic Analysis*, pages 11–17, 2007.
- [7] A. Lienhard, T. Gırba, and O. Nierstrasz. Practical object-oriented back-in-time debugger. In *ECOOP*, pages 1592–615, 2008.
- [8] A. Lienhard, O. Greevy, and O. Nierstrasz. Tracking objects to detect feature dependencies. In *International Conference on Program Comprehension*, pages 59–68. IEEE, 2007.
- [9] T. Richner and Stéphane Ducasse. Using dynamic information for the iterative recovery of collaborations and role. In *International Conference on Software Maintenance*, pages 34–43. IEEE, 2002.
- [10] N. Wilde and R. Huitt. Maintenance support for object-oriented programs. *IEEE Transactions on Software Engineering*, 18(12):1038–1044, December 1992.
- [11] R. Wirfs-Brock, B. Wilkerson, and L. Wiener. *Designing Object-Oriented Software*. Prentice Hall, 1990.

# Arabic Lisp

Hanan Elazhary  
Electronics Research Institute  
Cairo, Egypt  
[hanan@eri.sci.eg](mailto:hanan@eri.sci.eg)

**Abstract**— High-level languages and very high-level languages have been developed to simplify programming. However, programming is still not an easy task for everyone. This paper discusses the concept of facile programming, where programming languages should be made as easy as possible to be easily learnt, remembered, and used by people from different disciplines. This is achieved by trying to figure out *all* the difficulties that face such programmers and trying to tackle them. The paper addresses one difficulty, which is that not every programmer is given the chance to program in his natural language or at least a language like his natural language. Thus, it is hard for programmers who do not speak English to learn, remember, and use English-like programming languages and understand the English compilation error messages. Unfortunately, there are very few attempts in the literature to address this problem and these attempts do not handle common programming languages and thus, have not been accepted. To tackle this problem, we developed an Arabic version of Lisp, namely *Arabic Lisp*. We also developed *Arabic Lispizer*, which not only translates *Arabic Lisp* to Lisp (and vice versa for portability), but also detects syntax errors and produces corresponding error messages in Arabic.

## I. INTRODUCTION & BACKGROUND

IF two persons would like to communicate with each other, they have to speak the same language. If they do not speak the same language, one of them should learn the language of the other or they should use a translator. When a person attempts to speak a foreign language, in which he (he stands for he/she in the rest of the paper) is not fluent, he usually tends to think in his natural language and then translate his thoughts to the foreign language. This is why he usually makes some effort to remember the syntax of the foreign language and the sentences that he pronounces are not always grammatically correct. Besides, it is not always easy for him to understand this foreign language.

When the computer systems were invented, programmers had to learn the language of a computer system (machine language) in order to be able to communicate with it. Since machine language includes two letters only that are represented by zero volt and one volt, learning machine language is a very hard task. Besides, machine language is a low-level language that requires knowledge of the underlying hardware and is machine dependent. In other words, it differs from one family of computer systems to another. This means that programmers had to learn more

than one machine language. To avoid these problems, programmers had to use translators to translate programs from their natural languages to the required machine language. But, due to the ambiguity of natural languages [1], no such translators could be easily developed. For this reason, assembly language [2] had to be developed instead. Assembly language provides an English-like statement for each statement in machine language and assembly language programs are translated into machine language programs using assemblers. Unfortunately, assembly language is a low-level language and is machine dependent like machine language. Striving to make programming much easier, high-level languages and very high-level languages [3]-[6] have been invented. These languages are more user friendly since they are not machine dependent and do not require knowledge of the underlying hardware. Unfortunately, in spite of this, programming is still not an easy task for everyone especially programmers from disciplines that are not computer-related.

In this paper, we discuss the concept of facile programming, where programming languages should be made as easy as possible to be easily learnt, remembered, and used by people from different disciplines. The goal is to provide programmers from diverse backgrounds with programming capability. This is achieved by trying to figure out *all* the difficulties that face such programmers and trying to tackle them. What is noticeable about typical programming languages (assembly languages, high-level languages, and very high-level languages) is that they are mostly English-like. This is because they have been invented by English-speaking developers and have been intended to be used by English-speaking programmers [7]. Thus, these programming languages are not easily learnt, remembered, and used by programmers who do not speak English. Besides, it is very hard for them to understand compilation error messages. From the above discussion, it is obvious that if programming is to be easy for everyone, everyone should be given the chance to use a programming language that is like his natural language (or a natural language in which he is fluent). Developing such programming languages implies a faster learning phase, better remembrance of the programming language syntax, easier programming, less errors, and more programmers. There have been some attempts in the literature to develop programming languages that are like natural languages other

than English [8]-[12]. Similar attempts have been made for programming environments [13], [14]. But, to the best of our knowledge, such attempts have not been extended to cover common high-level and very high-level programming languages and thus these attempts have not been widely accepted. For example an Arabic programming language similar to C has been developed [15], but this language is not exactly like C. Thus, no translation is possible between these two languages and hence no portability is possible. Some other programming languages are partially English or English-like. For example, Microsoft Access [16] allows using the Arabic language in specifying names and data values in database systems. But, unfortunately, SQL (which is used to access database systems) keywords are English-like and error messages are English.

In an attempt to tackle this problem, we developed an Arabic version of Lisp [17], namely *Arabic Lisp*. The reason for selecting Lisp is that it is more suitable for the average novice programmers as explained in Section II. The reason for selecting the Arabic language is that it is the official language of hundreds of millions of people in the Middle East and North Africa. If an Arabic version of Lisp were developed, it is expected that a large number of these targeted users would be encouraged to learn it and use it.

This paper is organized as follows: Section II introduces Lisp while Section III introduces *Arabic Lisp* with examples. This section is well-explained such that even non-Arabic speakers can understand it. Finally, Section IV provides the conclusions and directions for future research.

## II. LISP

The Lisp language was first proposed in the late 1950s. Since then, many dialects of Lisp have been developed, but ANSI/INCITS 226-1994 (R2004) [18] is the official standard. The Common Lisp HyperSpec™ [19] is the acclaimed online version derived from this official standard. In Lisp, instructions and data are in the form of lists and thus it is called Lisp (List processing). Lisp is a general-purpose programming language that supports a combination of procedural, functional, and object-oriented programming paradigms. It does not normally require initial declarations needed in other general-purpose programming languages such as C++ and Java. Thus, Lisp is more suitable for the average novice programmers. Lisp uses the prefix notation (the function name usually precedes the arguments). Table I shows some basic Lisp instructions and examples of their operation.

## III. ARABIC LISP

In this section, we present *Arabic Lisp*, which is the Arabic version of Lisp. The *Arabic Lisp* keywords corresponding to the basic Lisp keywords are as follows:

- *setf*: ضع في
- *list*: كون قائمة من
- *append*: كون قائمة من مكونات

- *NIL*: فراغ (if it refers to an empty list) or خطأ (if it refers to "false")
- *push*: ادفع, *pop*: انزع من
- +: +, -: -, \*: \*, /: /
- The apostrophe ' : '
- *first*: اول
- *rest*: باقي
- *butlast*: انزع من نهاية, *last*: اخر
- *length*: طول
- *reverse*: اعكس
- *min*: اصغر, *max*: اكبر
- *expt*: اس, *sqrt*: جذر
- *equal*: يساوي
- *T*: صواب
- *member*: عضو
- *listp*: هل قائمة, *endp*: هل فارغة, *numberp*: هل زوجي, *zerop*: هل صفر, *oddp*: هل فردي, *plusp*: هل موجب, *minusp*: هل سالب
- *print*: اطبع
- *and*: و, *or*: او
- *if*: لو
- *defun*: عرف
- *apply* #: #طبق

Note that Lisp and *Arabic Lisp* use some similar keywords such as +, -, \*, and / and same Arabic digits (0, 1, 2, 3 ...). The *Arabic Lisp* instructions (corresponding to the Lisp instructions shown in Table I) are shown in Table II. In these examples, each Lisp keyword is replaced by the corresponding *Arabic Lisp* keyword. A character in the English alphabet (if not a part of a keyword) is replaced by its corresponding character in the Arabic alphabet. Accordingly, a is replaced by ا, b by ب, c by ت, d by د, e by ح, f by ف, g by ج, h by هـ, i by ي, j by ز, k by س, l by ل, m by م, n by ن, o by و, p by ط, q by ق, r by ر, s by ش, t by ث, u by هـ, v by و, w by و, x by خ, y by ي, z by ز. Note also that unlike English, Arabic is read from the right to the left. In example 1 in Table I, *setf* is replaced by *ضع في* and x is replaced by م. The instruction is then written from the right to the left as shown in example 1 in Table II. The reader can similarly understand how the other Lisp instructions in Table I can be replaced by the corresponding *Arabic Lisp* instructions in Table II. In examples 26 and 27, the name "addition" is replaced by the name *جمع* and the name "number\_list" is replaced by the name *قائمة ارقام*. There is no rule that applies to this since the names "addition" and "number\_list" are chosen by the Lisp programmer, while the name *جمع* and *قائمة ارقام* are chosen by the *Arabic Lisp* programmer.

We developed *Arabic Lispizer*, which can translate Lisp to *Arabic Lisp* and vice versa for portability and to be easily understood by *Arabic Lisp* programmers. One of the problems that faces *Arabic Lispizer* when translating *Arabic*



*Lisp* to *Lisp* (and vice versa) is the translation of names. For example when attempting to translate example ٢٦ in Table I to the corresponding example in Table II, it is not clear how to translate the name "addition" and the name "number\_list". To tackle this problem, as mentioned above, the character a is replaced by ا, b by ب, d by د, e by هـ, i by ي, l by ل, m by م, n by ن, o by و, r by ر, s by س, t by ت, and u by و. The name "addition" is thus translated into اضافة and the name "number\_list" is translated into قائمة. Note that in Arabic, the shapes of some characters change slightly when they are attached to form a word. Although these names are weird and meaningless in Arabic, they are acceptable since they are only names. The *Arabic Lisp* programmer can simply redefine the function using a suitable name of his choice without having to rewrite it.

*Arabic Lispizer* can also detect syntax errors, and produce corresponding error messages in Arabic to be easily understood by *Arabic Lisp* programmers. Table III shows some example *Arabic Lisp* errors, the corresponding error messages detected by *Arabic Lispizer* (in Arabic), and their translation:

- In example 1, the instruction وضع is unrecognized since وضع في, that corresponds to *setf* in Lisp is expected
- In example 2, the variable ب should be bound possibly by writing (وضع في ب فراغ), which corresponds to (*setf* b NIL) in Lisp, before attempting to push anything to it
- In example 3, a variable name should follow وضع في since in Lisp, a variable name follows *setf*
- In example 4, the instruction جذر, which corresponds to *sqrt* in Lisp, expects a number argument. Thus, the provided argument type is unacceptable

#### IV. CONCLUSIONS & FUTURE WORK

This paper discussed the concept of facile programming, where programming languages should be made as easy as possible to be easily learnt, remembered, and used by people from different disciplines. This is achieved by trying to figure out *all* the difficulties that face such programmers and trying to tackle them. What is noticeable about typical programming languages (assembly languages, high-level languages, and very high-level languages) is that they are mostly English-like. This is because they have been invented by English-speaking developers and have been intended to be used by English-speaking programmers. Thus, these programming languages are not easily learnt, remembered, and used by programmers who do not speak English. Besides, it is very hard for them to understand English compilation error messages. In an attempt to tackle

this problem, we developed an Arabic version of Lisp, namely *Arabic Lisp*. *Arabic Lisp* covers the basic Lisp instructions, but due to the space limitations of the current paper, not all instructions are shown. We also developed *Arabic Lispizer*, which not only translates *Arabic Lisp* to Lisp (and vice versa for portability), but also detects syntax errors and produces corresponding error messages in Arabic.

As a future work, we intend to extend *Arabic Lisp* to cover all Lisp instructions. *Arabic Lispizer* should be also extended to cover all possible syntax errors so as not to leave them for Lisp compiler that produces error messages in English. We also intend to develop the Arabic versions of more programming languages to attract more Arabic-speaking programmers. Finally, we intend to continue our work in facile programming by trying to study other programming difficulties and problems and trying to tackle them. Our aim is to make programming easy and facile for everyone regardless of their background.

#### REFERENCES

- [1] D. Jurafsky and J. Martin, *Speech and Language Processing*, 2nd ed., Prentice Hall, 2008.
- [2] R. Hyde, *The Art of Assembly Language*, No Starch Press, Sep. 2003.
- [3] R. Hyde, *Write Great Code*, 1st ed., No Starch Press, March 2006.
- [4] T. Gaddis, *Starting Out with C++: From Control Structures through Objects*, 6th ed., Addison Wesley, March 2008.
- [5] J. Bloch, *Effective Java*, 2nd ed., Prentice Hall, May 2008.
- [6] A. Beaulieu, *Learning SQL*, 1st ed., O'Reilly Media, August 2005.
- [7] [http://en.wikipedia.org/wiki/Non-English-based\\_programming\\_languages](http://en.wikipedia.org/wiki/Non-English-based_programming_languages)
- [8] W. Maojiang and D. Wei, "Suggestions for adding the ability of processing of Chinese characters in programming languages," *Computer Standards & Interfaces*, vol. 8, no. 2, 1988-1989.
- [9] H. Al-Sadoun, M. Yaseen, A. El-Jallad, and M. El-Jallad, "ARbic Basic (ARBI): A new Arabic MS-DOS based programming language," in *Proc. of the 12th National Computer Conference and Exhibition*, Oct. 1990.
- [10] T. Tamai, "On Japanese-based programming," *Journal of Information Processing Archive*, vol. 13, no. 1, pp. 49-56, 1990.
- [11] M. Amin, "The Arabic object-oriented programming language Al-Risalh," in *Proc. of ACS/IEEE Intl. Conference on Computer Systems and Applications*, June 2001, pp. 424-427.
- [12] I. Tetsuji, S. Hiroshi, O. Osamu, and U. Shunsuke, "Grammar of a Japanese-based programming language "Mahoroba"," *Joho Shori Gakkai Kenkyu Hokoku*, vol. 2001, no. 31-(SE-130), 2001.
- [13] A. Rafea, D. Soliman, E. Samy, and G. Felfela, "Al-Daleel: An Arabic interactive programming environment," in *Proc. of the 3rd Intl. Conf. and Exhibition on Multi-Lingual Computing*, 1992.
- [14] A. Al-Salman, "An Arabic programming environment," in *Proc. of the 1996 ACM Symposium on Applied Computing*, 1996.
- [15] <http://www.jeemlang.com/documentation/webframe.html>
- [16] J. Viescas and J. Conrad, *Microsoft Office Access Inside Out*, Microsoft Press, April 2007.
- [17] P. Seibel, *Practical Common Lisp*, APress, 2005.
- [18] <http://www.lispworks.com/documentation/>
- [19] <http://www.lispworks.com/>

TABLE I  
BASIC LISP INSTRUCTIONS AND EXAMPLES OF THEIR  
OPERATION

No.	Lisp instruction	Result
1	(setf x 5)	5
2	(setf x (list x x))	(5 5)
3	(setf x (append x x))	(5 5 5 5)
4	(setf y NIL)	NIL
5	(push 8 y)	(8)
6	(+ 5 7)	12
7	(first '(a b c d))	A
8	(rest '(a b c d))	(B C D)
9	(butlast '(a b c d e f) 3)	(A B C)
10	(last '(a b c d e))	(E)
11	(length '(a b))	2
12	(reverse '(a b))	(B A)
13	(min 3 5 7 1)	1
14	(expt 2 4)	16
15	(sqrt 9)	3
16	(equal (* 3 3) 9)	T
17	(member 3 '(1 2 4))	NIL
18	(listp '(a b c))	T
19	(evenp 4)	T
20	(plusp 8)	T
21	(boundp 'c)	NIL
22	(setf c 5) (print c)	5 5
23	(and (zerop 0) (plusp -6))	NIL
24	(or (zerop 0) (plusp -6))	T
25	(if (= 4 (+ 3 1)) 7 8)	7
26	(defun addition (number_list) (apply #'+ number_list))	ADDITION
27	(addition '(3 4 5))	12

TABLE II  
ARABIC LISP INSTRUCTIONS CORRESPONDING TO THE LISP  
INSTRUCTIONS OF TABLE I

No.	Arabic Lisp instruction	Result
1	(ضع في م 5)	5
2	(ضع في م (كون قائمة من م م))	(5 5)
3	(ضع في م (كون قائمة من مكونات م م))	(5 5 5 5)
4	(ضع في ن فراغ)	فراغ
5	(ادفع 8 ن)	(8)
6	(7 5 +)	12
7	(اول '(ا ب ت ث))	ا
8	(باقي '(ا ب ت ث))	(ب ت ث)
9	(انزع من نهاية '(ا ب ت ث ج ح) 3)	(ا ب ت)
10	(اخر '(ا ب ت ث ج))	(ج)
11	(طول '(ا ب))	2
12	(اعكس '(ا ب))	(ب ا)
13	(اصغر 1 7 5 3)	1
14	(اس 4 2)	16
15	(جذر 9)	3
16	(يساوي (* 3 3) 9)	صواب
17	(عضو 3 '(4 2 1))	خطأ
18	(هل_قائمة '(ا ب ت))	صواب
19	(هل_زوجي 4)	صواب
20	(هل_موجب 8)	صواب
21	(هل_معرف 'ت)	خطأ
22	(ضع في ت 5) (اطبع ت)	5 5
23	(و(هل_صفر 0) (هل_موجب -6))	خطأ
24	(او(هل_صفر 0) (هل_موجب -6))	صواب
25	(لو (= 4 (+ 3 1)) 7 8)	7
26	(عرف جمع (قائمة_ارقام) (طبق #' + قائمة_ارقام))	جمع
27	(جمع '(5 4 3))	12

TABLE III  
EXAMPLES OF ARABIC LISP ERRORS, THE CORRESPONDING ERROR MESSAGES IN ARABIC, AND THEIR TRANSLATION

No.	Arabic Lisp instruction	Error message	Translation
1	(ضع 5 في س)	أمر ضع غير مفهوم	The instruction ضع is unrecognized
2	(ادفع 8 ب)	المتغير ب غير معرف	The variable ب is unbound
3	(ضع في (كون قائمة من س س))	هناك متغير ناقص	A variable is missing
4	(جذر (9))	نوع المعامل غير مقبول	The argument type is unacceptable

# The use of reading technique and visualization for program understanding

Daniel Porto  
Federal University of São Carlos  
São Carlos - Brazil  
daniel\_porto@dc.ufscar.br

Manoel Mendonça  
Federal University of Bahia  
Salvador - Brazil  
manoel.g.mendonca@gmail.com

Sandra Fabbri  
Federal University of São Carlos  
São Carlos - Brazil  
sfabbri@dc.ufscar.br

## Abstract

*Code comprehension is the basis for many other activities in software engineering. It is also time consuming and can greatly profit from tools that decrease the time that it usually consumes. This paper presents a tool named CRISTA that supports code comprehension through the application of Stepwise Abstraction. It uses a visual metaphor to represent the code and supports essential tasks for code reading, inspection and documentation. Three case studies were carried out to evaluate the tool with respect to usability and usefulness. In all of them the experiment participants considered that the tool facilitates code comprehension, inspection and documentation.*

## 1. Introduction

Code comprehension supports key software engineering activities like maintenance and inspection [1]. Software maintenance, the most expensive and time consuming activity in large software systems, is heavily based on software comprehension. Neginhal and Kothari [2] estimate that software engineers spend approximately 90% of the maintenance time trying to understand the program. Likewise software inspection is heavily dependent on the inspector ability to understand the actual program behavior in relation to the expected one [3].

With the gradual increase in the complexity and size of the software currently developed, software comprehension presents itself as a growing challenge for software engineers. For this reason, different approaches to facilitate the comprehension are being investigated. Among those, one can mention software reading and visualization techniques.

Reading techniques are a set of steps that supports the individual analysis of a specific artifact to execute a specific task [4]. Similarly, visualization techniques can facilitate the understanding and identification of

information contained in software artifacts.

Another way of facilitating software comprehension is through the use of automation. Tilley, Paul and Smith [5] comment that software tools are very important to help decrease the inherent complexity of the comprehension process.

Considering this scenario, we developed a tool named CRISTA (Code Reading Implemented with Stepwise Abstraction) to support code comprehension through a visual metaphor and the application reading by Stepwise Abstraction (SA) technique [6].

This paper shows how CRISTA supports code comprehension and how it can be applied during inspection and code documentation activities in a systematic way. It is organized as follows: Section 2 presents concepts of code comprehension, reading techniques and code visualization. Section 3 presents CRISTA tool describing its main characteristics for code comprehension supporting. Section 4 comments the case studies that were carried out to evaluate the use of the tool. Section 5 presents the conclusion and ongoing works.

## 2. Code Comprehension

According to Vinz and Etzkorn [7], software comprehension refers to any activity that uses static or dynamic methods to reveal software properties.

Software comprehension supports software engineering activities such as testing, inspection, maintenance and reengineering [1]. Hence, good software comprehension is needed to successfully execute these activities and, consequently, to engineer and evolve software systems.

Due to its importance several alternatives have been proposed to improve and facilitate software comprehension. Good engineering practices such as standardization, simplicity, good documentation and encapsulation are good ways to facilitate software comprehension. However, we are frequently faced with legacy systems with poor documentation and

structure, many times with nothing more than the source code. Code comprehension is needed and it is an expensive and complex endeavor in such cases [8].

One needs systematic code comprehension approaches, especially for parts of the code where the required cognitive complexity is high [3]. Code reading is a systematic approach for code comprehension that uses explicit and systematic procedures to understand the code.

According to O'Brien [9], there are three strategies for systematic code comprehension: top-down, bottom-up and a hybrid strategy. The top-down strategy suggests that the programmer uses the knowledge of the domain to construct a set of expectations that would be expected in the code. The bottom-up strategy works from the ground up. Small code blocks are read and abstracted into more generic and larger blocks. The hybrid strategy combines both. Our work uses reading by SA [6] a bottom-up strategy.

## 2.1. Reading by Stepwise Abstraction

Stepwise Abstraction is a reading technique that supports code comprehension and is used in code inspection process to help defect detection [6]. Several experiments that investigate the effectiveness of testing and inspection activities use this technique for code inspection [10]. The code functionalities are determined through abstractions obtained from the code by reading it from its internal to its external structures. The reader writes down his understanding of each code block, building bottom-up his comprehension of the code as a whole.

If the technique is being applied in an inspection activity for example, the inspector can compare his abstractions with the original code specification. The goal in this case is to identify inconsistencies between the required and the implemented functionalities.

SA is well known by the software inspection community, but it is not as widespread in other software engineering areas. However, it is a systematic way to understand the code and, as such, can be used in any other activity in which code understanding is needed.

## 2.2. Code Visualization

There are several models proposed in the literature to understand the code, but regardless of the type considered, the comprehension is always related to the transfer and building of knowledge about the software, from the artifacts available for analysis [11].

According to Tergan and Keller [12], the cognitive process of human beings is more intuitive, effective

and efficient when supported by visual resources such as images, diagrams and signs. Thus, information visualization can be an important resource for software comprehension. It can support the representation, and knowledge transfer, of the key aspects of software artifacts under analysis.

Visual representations of software artifacts are already used as enablers of software understanding in several areas of software engineering. Greevy, Lanza and Wyseier [13], for example, emphasize that reverse engineering needs high level abstractions and views that represents different aspects of the software.

The growth of the size and complexity of software systems has fostered the creation of software visualization techniques that have introduced many alternative ways to facilitate software understanding.

Code visualization is the research area that investigates ways to represent, in a graphical format, key aspects of the source code, to facilitate its understanding.

The CRISTA tool, presented in the next section, uses this principle. It represents the source code blocks as a set of nested rectangles that facilitates the execution of the SA reading technique, as explained in the following section.

## 3. CRISTA

The Code Reading Implemented with Stepwise Abstraction (CRISTA) tool was originally implemented to aid the inspection of Java source code. However, the tool can be used to aid any source code comprehension activity.

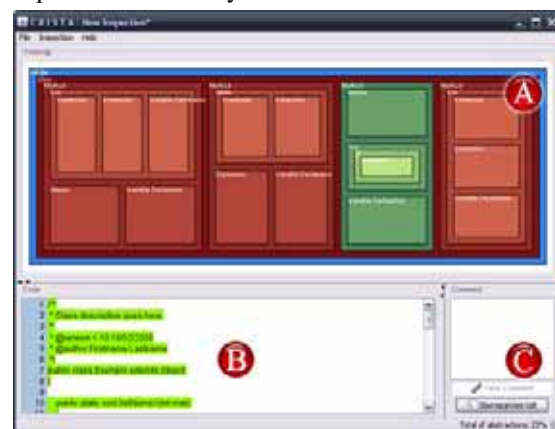


Figure 1. CRISTA main screen

As seen in Figure 1, CRISTA adopts a visual metaphor that represents visually the hierarchical blocks of the code being analyzed (Area A). The visual metaphor, called Treemap [14], offers a simply way to look at the code structure. It separates code blocks as

nested rectangles in accordance with their hierarchy. It also works as a visual feedback for the code analysis process, changing the rectangle colors from red to green as the reader documents his abstraction of the blocks under analysis.

The actual code is shown on the bottom left side (Area B) of Figure 1. This window can be browsed and it is logically linked to the visualization area. Any rectangle clicked on the visual screen highlights the corresponding block in the source code window and vice-versa.

Once a code block is selected, its abstraction can be entered, as free text, in the documentation area (Area C). This comment will be physically associated to the selected block, and can have a later use to produce code documentation or pseudo-code, as discussed ahead. Any time comment is associated with a code block, the rectangle corresponding to this block turns green. This allows the reader to easily follow the progress of the code comprehension process.

By default, the tool enforces reading by SA. In other words, an external code block can only be documented if all its internal blocks were documented first. This procedure can be broken if the reader explicitly turning off the SA option in the tool menu.

The treemap visual metaphor is very efficient in terms of space usage. It can represent very large collections of nested rectangles. This way the reader can visualize large pieces of code in just one screen. The current version of CRISTA can represent up to ten hierarchical levels in one screen and can navigate across larger hierarchies. This allows the tool to tackle virtually any size of source code.

### 3.1. Using CRISTA to Document and Reverse Engineer the Source Code

Once a few abstractions have been entered, CRISTA permits to export files in which documented parts of the source code is substituted with its abstraction. Figure 2 shows an example of this functionality in a piece of code. In it, the highlighted line was already abstracted while the rest of the printing is raw source code. This printing helps to synthesize what was already abstracted and can be used to facilitate the understanding of what is yet to be abstracted.

```

18 public static void listnums2(int max)
19 {
20     int i = 0;
21     while i is less than max, print i and increment it.
22     system.out.printf("end");
23 }

```

**Figure 2. Code substituted by abstractions**

Once the whole code has been abstracted, the tool can export a file with all abstractions hierarchically

organized. If the abstraction is written in “structured English”, this file will look like an algorithm pseudo code, as seen in Figure 3. This type of structure maybe facilitates comprehension in activities like software maintenance and reengineering.

```

public static int min(int... numbers)
{
    Initiate the less number with the greater integer values that is possible
    For each number of the list
        If the current number is less than the least that was found
            {
                Store the new minimal value.
            }
    }
    Return the new minimal value.
}

```

**Figure 3. Abstractions as algorithm**

In a similar fashion, CRISTA permits to export the source code together with the user abstractions. In this case, the tool exports the abstraction as source code comments. It is possible to choose the types of blocks in which the user wants to insert the comments. For example, the user can put a comment for all methods of the source code. This functionality can thus be used to systematically re-document the abstracted code.

Another way to present the abstractions is by means of a functional description report. This option generates a HTML document that contains a description of the code structures selected by the user. In the case of Figure 4, methods were selected as re-documentation targets. The report contains the method signature, followed by its abstraction. Moreover, the code can be seen by clicking on the “+” signs under its abstraction.

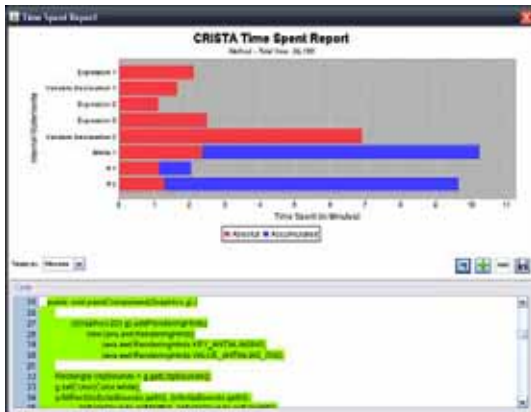


**Figure 4. Functional description**

### 3.2. Data Gathering

As code comprehension is a time consuming activity, it is important that the tool gathers a rich data set about the time consumed in this activity. CRISTA collects effort data as the user execute each abstraction activity. It systematically records the elapsed time between the selection of a code block and the completion of its description. This information can then be used to analyze which pieces of code was most

time consuming to comprehend. CRISTA shows the time information in the graphical format (Figure 5).



**Figure 5. Time spent to abstract a Class**

Figure 5 example shows the time associated with a method of a class. Each bar represents a code block and is split in two colors. The first color (red) indicates the time spent to abstract only the block structure. The second color (blue) indicates the time spent to abstract the code internal structures.

Clicking on a bar of the diagram of Figure 5, it details the time spent of internal instructions of the corresponding block. A new graph is shown in the same way as in Figure 5.

The data gathered by CRISTA helps to identify which are the more complex (or less readable) pieces of code. Software engineers can use this information to identify code blocks that need preventive maintenance, reengineering, or detailed re-documentation.

### 3.3. Code Inspections

Aiming to support code inspection, CRISTA allows the reader to register code discrepancies. The user can enter a new discrepancy and mark any piece of code he wants to associate with it.

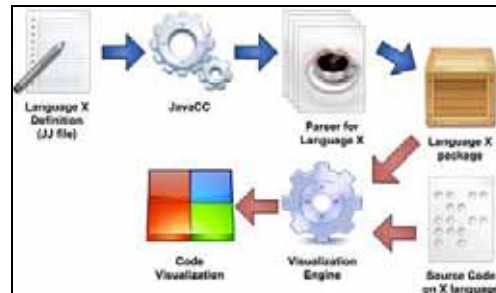
All lists of discrepancies can be saved and the tool supports the merging of discrepancies list in order to aid on inspection group meetings. The tool does that by keeping a *final* discrepancy list. This list is assembled by adding discrepancies from all inspector lists.

The tool can produce discrepancies reports that list all discrepancies and associated code blocks found in the final list.

### 3.4. CRISTA architecture

Figure 6 shows the works of all CRISTA components. The tool was implemented as a desktop application in order to facilitate individual off network

usage. Data persistence is done using a file rather than databases. It is implemented in Java and does not need other associated applications. This makes CRISTA quite portable and platform independent.



**Figure 6. CRISTA architecture**

The use of a file to save abstractions and discrepancies facilitates the exchange of information between software engineers. In the case of inspections, for example, all discrepancy lists can be easily sent to the inspection moderator.

The CRISTA visualization engine works over an abstract syntax tree built by parsing the code under analysis. JavaCC [15] is used to create the code parser. Currently the tool supports Java and C++. A Cobol parser is also being currently developed. The following steps should be executed to add a new language capability to CRISTA:

- 1) Build the language parser in the JavaCC format.
- 2) Use JavaCC to produce the parser as Java libraries.
- 3) Bundle this library with the CRISTA executable.

The tool uses the Model, View, Controller (MVC) pattern to separate the persistence, interface and business rule layers. The interface layer is implemented using Java Swing and the libraries Jwizard [16] and JavaHelp [17] to build the wizard and help functionalities respectively.

CRISTA uses the Prefuse visualization framework to create the treemap views [18]. Prefuse is freely available and it is Java compliant.

The interested reader can download a video that demonstrates CRISTA main capabilities from the following web address:

[http://www.dc.ufscar.br/~daniel\\_porto/Crista\\_ing.wmv](http://www.dc.ufscar.br/~daniel_porto/Crista_ing.wmv)

## 4. Evaluating CRISTA

We executed three case studies to evaluate CRISTA functionalities as well as its benefits, in the context of code comprehension and inspection.

In all cases the study involved groups of students that were trained and prompted to execute a set of tasks using CRISTA. At the conclusion of the tasks the students were asked to fill out a questionnaire

composed of the following sections: i) characterization of the study participant profiles with respect to experience and background; ii) characterization of the training received by the participants; iii) subjective evaluation of the tool usability using Nielsen-Molich's heuristics [19]; iv) subjective evaluation of tool functionalities with respect to efficiency and effectiveness; v) subjective evaluation of usefulness of the reports produced by the tool; and vi) space for freely commenting on the tool's weaknesses and strengths.

#### 4.1. Case Study 1

The goal of the first study was to evaluate CRISTA main functionalities and its usability. Twenty junior undergraduate students from the Federal University of São Carlos (UFSCar) in Brazil participated in this study. They were taking the Software Engineering course and knew all basic concepts of software inspection and Java.

The study was broken down in three parts. First, we trained the participants. This involved training on reading by SA as well as the use of the Nielsen-Molich's heuristics for software usability [19]. The students were then given an exercise to apply SA by hand on a small piece of code with about 60 lines.

The second part consisted of using the CRISTA tool for code inspection and was executed on the following week. The students had to follow the SA technique using the tool but did not receive any tool training. They were given two hours to inspect two pieces of code with 100 lines each. After this, they were asked to answer the feedback questionnaire. Besides, a brainstorm about the tool was conducted and the doubts were solved.

The third part was executed during the following two weeks. Students were split into two groups and given a piece of 300 lines of code to review. The first group inspected the code by hand and the second group used CRISTA. The following week the students received another piece of code with the same size and the procedure was inverted, group 1 used CRISTA and group 2 did the inspection by hand. After the two week effort, the students were asked to answer the feedback questionnaire once again.

The collected data indicated that the students spent the same time and found a similar number of discrepancies on the code, when doing the review by hand and using CRISTA. The time to produce the final list of discrepancies was smaller with CRISTA. The subjective feedback indicated that the students found the tool intuitive and easy to use.

#### 4.2. Case Studies 2 and 3

Case Studies 2 and 3 had the objective of evaluate tool usability as well as its influence on the application of the SA technique. The participants were seven graduate students from UFSCar (case study 2) and fourteen graduate students from the University of São Paulo, USP (case study 3).

Both studies followed the same design. Students were trained in SA and in the CRISTA tool. Immediately afterwards, they did a procedure identical to part three of the first case study. They inspected the same two pieces of code using CRISTA and inspection by hand in a two week period. Likewise, they answered the feedback questionnaire after these two inspections.

Results were similar to the first case study. Feedback on all studies was gathered to improve CRISTA. Students suggested that the tool would benefit from a code search facility. This functionality was added to the tool as a result of that.

### 5. Conclusion

Code comprehension is an important activity in software engineering. It has a strong impact in other activities such as code inspection, reengineering, reverse engineering and maintenance.

Code comprehension is difficult and complex in large legacy systems and can benefit strongly from tools that support systematic code analysis.

This article presented CRISTA, a tool that supports code comprehension by the use of SA and visual metaphors. The tool produces several reports that facilitate code comprehension. It generates data reports that permit the identification of the most effort prone code blocks. It helps code documentation and supports the recording of code discrepancies as well as the composition of code discrepancy lists.

The tool was projected so that it can integrate new language parsers. This way it can be continuously expanded to support new programming languages. The standardized and easy to use interface can serve as a common code comprehension reference in companies that use several languages to develop software. Currently, CRISTA works with Java and C++. A version that also supports Cobol is being finalized [20].

The tool was evaluated in three case studies with group students (41 students in total). The studies indicate that the tool is easy to use and can systematize code comprehension and documentation. The tool presented similar results on discrepancy detection efficiency and efficacy when compared to inspections

done by hand, with a few gains on time in the compilation of discrepancies lists. However, the studies indicate that the tool usefulness grows with the size and complexity of the analyzed code.

As future work, we intend to run more realistic experiments with the tool. We want to evaluate it with larger pieces of code, preferably in industrial settings. This will allow us to look at the tool performance on real world software comprehension activities. Besides, the tool only allows loading one single file each time. Hence, we plan to implement the treatment of higher level of blocks, such as packages, or provide links to other files that allow understanding more complex software interactions. Another point to be considered is how to merge comments from different readers.

We also want to test the tool performance on different software maintenance activities. This will help us to evolve the tool with respect to functionalities to support those new activities, to improve visualization resources and to expand data gathering capabilities.

We also intend to explore different forms of systematic code analysis, ranging from well structured SA to free ad-hoc code abstraction. By doing that, we may be able to identify and evaluate new strategies for code comprehension.

## Acknowledge

We thank the financial support from the Brazilian agencies CAPES and CNPq

## References

- [1] N. Walkinshaw, M. Roper, and M. Wood, "Understanding object-oriented source code from the behavioural perspective." in Proceedings of IEEE International Workshop on Program Comprehension pp. 215-224, 2005.
- [2] S. Neginhal, and S. Kothari, "Event Views and Graph Reductions for Understanding System Level C Code," in Proceedings of the 22nd IEEE International Conference on Software Maintenance, pp. 279-288, 2006.
- [3] P. Runeson, and A. Andrews, "Detection or isolation of defects? An experimental comparison of unit testing and code inspection." in Proceedings of 14th International Symposium on Software Reliability Engineering pp. 3-13, 2003.
- [4] V. R. Basili, S. Green, O. Laitenberger *et al.*, "The empirical investigation of Perspective-Based Reading," *Empirical Software Engineering*, vol. 1, no. 2, pp. 133-164, 1996.
- [5] S. R. Tilley, S. Paul, and D. B. Smith, "Towards a framework for program understanding," in Proceedings of Program Comprehension, pp. 19-28, 1996.
- [6] R. C. Linger, H. D. Mills, and B. I. Witt, *Structured Programming: Theory and Practice*: Addison-Wesley, 418 p. 1979.
- [7] B. L. Vinz, and L. H. Etzkorn, "A Synergistic Approach to Program Comprehension." in Proceedings of IEEE International Conference on Program Comprehension. pp. 69-73, 2006.
- [8] J. Rilling, and T. Klemola, "Identifying comprehension bottlenecks using program slicing and cognitive complexity metrics." in Proceedings of 11th IEEE International Workshop on Program Comprehension pp. 115-124, 2003.
- [9] M. P. O'Brien, *Software Comprehension – A Review & Research Direction*, Technical Report UL-CSIS-03-3 UL-CSIS-03-3, University of Limerick, Limerick, 2003.
- [10] V. Basili, G. Caldiera, and F. Shull, "Studies on reading techniques," in Proceedings of the Twenty-First Annual Software Engineering Workshop, pp. 96-102, 1996.
- [11] S. Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*: Springer-Verlag, New York, Inc., 187 p. 2007.
- [12] S. Tergan, and T. Keller, *Knowledge and Information Visualization: Searching for Synergies*: Springer, 385 p. 2005.
- [13] O. Greevy, M. Lanza, and C. Wyseier, "Visualizing live software systems in 3D," in proceedings of the 2006 ACM symposium on Software visualization, New York, NY, USA, pp. 47-56, 2006.
- [14] B. Johnson, and B. Shneiderman, "Tree-maps: A space-filling approach to the visualization of hierarchical information structures." in proceedings of IEEE Visualization Conference, pp. 284 - 291, 1991.
- [15] JAVACC. "Java Compiler Compiler," 17/04/2008; <https://javacc.dev.java.net>.
- [16] JWIZARD. "JWizard," 17/04/2008; <http://flib.sourceforge.net/JWizard/doc>.
- [17] JAVAHELP. "JavaHelp System," 17/04/2008; <http://java.sun.com/products/javahelp>.
- [18] PREFUSE. "The Prefuse Visualization Toolkit," 17/04/2008; <http://prefuse.org>.
- [19] J. Nielsen, and R. Molich, "Heuristic evaluation of user interfaces," in Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people, Seattle, Washington, United States, pp. 249-256, 1990.
- [20] D. P. Porto, M. Mendonça, and S. C. P. F. Fabbri, "CRISTA - Code Reading Implemented with Stepwise Abstraction." in proceedings of Simpósio Brasileiro de Engenharia de Software - Sessão de Ferramentas, 6 p. 2008.



# Language Support for Event-Based Debugging

Ziad Al-Sharif, Clinton Jeffery  
Computer Science Department  
University of Idaho  
zsharif@ieee.org, jeffery@cs.uidaho.edu

## Abstract

*An event-based debugging framework provides high level facilities for debuggers that observe, monitor, control, and change the state and behavior of a buggy program. This paper introduces a set of additions to the Unicon programming language that enables debuggers to be written at a high level of abstraction. The extensions provide in-process debugging support with simple communication and no intrusion on the buggy program space. These language extensions have been tested and refined within a multi-agent debugging architecture called IDEA, and an extensible source-level debugger called UDB.*

## 1. Introduction

Unicon [7] is an object-oriented dialect of Icon [3], a very high level programming language with dynamic and polymorphic structure types, along with generators and goal-directed evaluation. Historically, the Icon language community had no formal debugging tool, only a trace facility. A rationale for this was that the very high level nature of Icon reduces the need for conventional debugging because programs are shorter. However, Unicon programs are often much larger than was common for Icon, and very high level languages' advanced features may introduce special kinds of bugs and create special needs for debugging tools.

Icon and Unicon are supported by the Alamo monitoring framework [6,8,9]. Alamo was originally designed for passive observation of program execution, suitable for software visualization tools but not adequate for a source-level debugger. This paper presents recent extensions that enable Alamo to serve as a debugging framework. The result of these extensions is called AlamoDE (Alamo Debug Enabled). AlamoDE is a framework that 1) includes debugging-oriented virtual machine instrumentation, 2) supports additional execution state inspection and source code navigation, 3) provides debugging tools with the ability to change the execution state by safely assigning to a buggy program's variables and procedures, and most importantly, 4) facilitates on the fly debugging extensions and cooperation. Different AlamoDE-based debugging tools can be written and tested as standalone programs and then loaded into a debugger to work in concert with each other.

This paper presents the design and implementation of AlamoDE and shows the novelty of its debugging support. Section 2 introduces Alamo's features that are ideal for debugging needs. Section 3 discusses AlamoDE, and its support for classical and advanced debugging techniques. Section 4 describes technical issues and features added, in both the Unicon virtual machine and the Alamo framework, to implement different functionalities for debugging reasons. Section 5 provides an evaluation, and introduces a source-level debugger that utilizes AlamoDE's features. Section 6 compares Unicon's debugging support with related work. Section 7 highlights planned future work. The conclusion from our experiment is in Section 8.

## 2. Background

Alamo is a lightweight architecture for monitoring developed originally to support program visualization. Alamo provides the Unicon virtual machine with monitoring facilities, and the capability for a program to load another program and execute it in a controlled environment.

The Alamo architecture is based on the thread model of execution monitoring, where a monitor program and the monitored program are in separate threads in a shared address space. Unicon's threads are called *co-expressions*. A co-expression is a synchronous thread inside the virtual machine. The evaluation of a co-expression requires both an interpreter stack and a C stack that are separate from the stacks of the main program. Alamo extended the Unicon co-expression facility with the ability to load a program, and sets it up with its own code, static data, stack, and heap, without linking symbols into the current program. This execution model provides no intrusion on the monitored program space, which is ideal for classical debugging, and simplifies the process of extending a debugger with external standalone debugging and visualization tools.

Switching between any two co-expressions is done through a small piece of assembler code that performs a lightweight context switch. The state of the program is saved and the control is transferred into the other program without the involvement of the operating system. Because they are synchronous, co-expression switches are much faster than typical thread switches such as those provided by the *pthread*s library.

### 3. AlamoDE

Many debugging architectures are based on inter-process communication which is good for remote debugging, but imposes an extra layer of operating system overhead in other contexts. Based on Alamo, AlamoDE provides an in-process debugging architecture with modest communication overhead. AlamoDE debugging support is provided through execution events and high level built-in functions that allow a debugging tool to inspect, change, observe, analyze, and control the state and behavior of the buggy program.

AlamoDE's goals include: 1) the ability to write debugging tools at a high level of abstraction, 2) all the usual capabilities of classical debuggers, 3) support for the creation of advanced debugging features such as automatic debugging, and dynamic analysis techniques, 4) the ability to debug novel language features such as generators, goal-directed evaluation, and string scanning, and 5) extensibility that allows different standalone debugging tools to work in concert with each other.

#### 3.1. Debugging Events

Considering the many millions of events produced by Alamo's detailed VM instrumentation, which provides 118 kinds of events, an efficient filtering mechanism is needed. Alamo used a simple bit vector called an *event mask* to specify event types of interest. For AlamoDE, the filtering was extended so that each event type of interest could have an associated *value mask*, a set of event values of interest which further restricts whether an event is reported; see Figure 1. Both the event mask and value mask are dynamic. This allows a debugging tool to change and customize the monitored events on the fly during the course of execution; any change on either of the two masks will immediately change the set of prospective events.

For example, placing a breakpoint on one or more line numbers requires the `E_Line` event to be part of the event mask. The value mask provides the ability to limit the reported `E_Line` events to those line numbers that have breakpoints on them. To clear a breakpoint, a tool removes the line number from the value mask. The `E_Line` event is removed from the event mask only if there are no more breakpoints and no other requests for `E_Line` events by the debugging core or any of its cooperative tools.

Even though debugging and visualization serve many common goals, for AlamoDE, the underlying instrumentation was extended with two additional event types that are needed for debugging. The new events are: 1) `E_Deref` reports when a variable is read (dereferenced). This event is needed to implement watchpoints on specific variable(s), and 2) `E_Syntax` reports when a major syntax construct such as a loop starts or ends. This event was inspired by the needs of automatic debugging systems [1,2] and required that syntax information be added to the

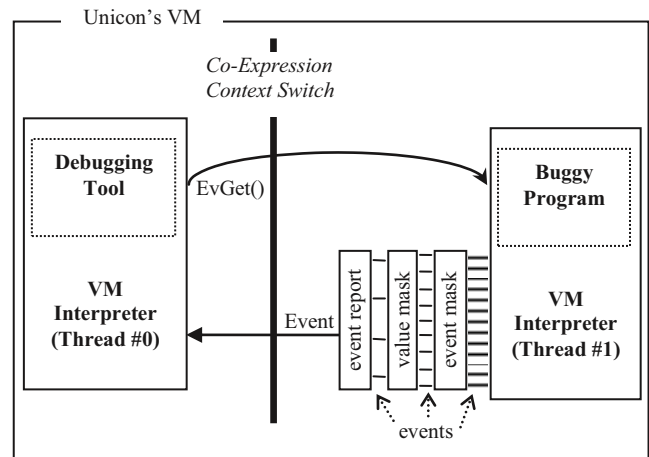


Figure 1. AlamoDE Architecture

Unicon virtual machine bytecode executable format. See Section 4.3 for more syntax instrumentation details.

#### 3.2. Debugging Functions

The Unicon language provides some reserved global names prefixed with ampersand (&) called *keywords*. Some keywords are introduced by Alamo for monitoring needs. For example, `&eventsource` contains a reference to the currently monitored program. Other keywords are used for error reporting and debugging. For example, the keywords `&file`, `&line`, `&column`, and `&syntax` report the currently executed file name, line number, column number, and syntax name respectively. These keywords can be inserted directly in the source code of the buggy program for debugging with print statements and assertions.

These keywords are made accessible to debugging tools using the `keyword()` built-in function, which is used to look up a value of a keyword in the buggy program. For example, `keyword("&file",&eventsource,0)` returns the name of the source file that contains the call statement, which instantiated the activation record currently at the top of the buggy program's call stack. Similarly, `keyword("&line",&eventsource,5)` looks up the buggy program's call stack, and returns the line number of the statement for which the fifth outer most activation record was instantiated.

Likewise, AlamoDE utilizes a set of functions, some of which are needed for observing, inspecting, changing, and controlling the state of the buggy program, while others are needed to support the ability to employ and incorporate external standalone debugging tools into another tool. For example, `EvInit()` loads and initializes the buggy program under the debugging tool, `EvGet()` starts/resumes the execution of the buggy program and installs/changes the set of requested events, and `EvSend()` forwards an event's code and value from one debugging tool into another; this is mostly used to send the most recent event to an external debugging tool. See Figures 2 and 3.

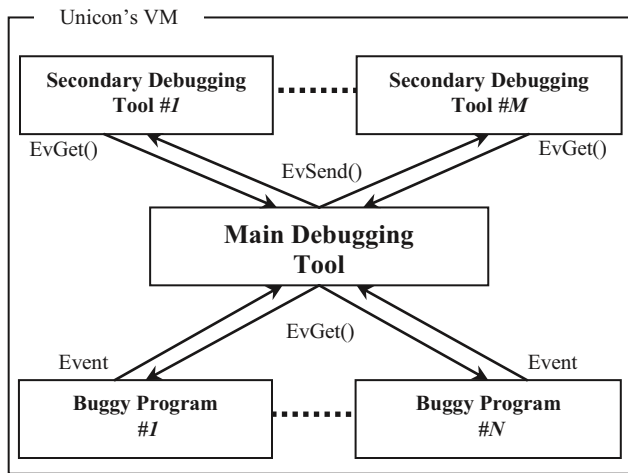


Figure 2. An AlamoDE debugging scenario

### 3.3. Controlling the Buggy Program

In AlamoDE, a debugging tool runs as the main co-expression inside the virtual machine. A buggy program and secondary standalone debugging tools can be loaded into different co-expressions controlled by the debugger. A debugger transfers control to the buggy program using the `EvGet()` function, which performs a lightweight context switch. After a context switch, the buggy program executes at full speed until there is some event that is of interest to the debugger. Instrumentation in the virtual machine reports an interesting event in the buggy program execution to the debugger by performing a context switch. The return value from `EvGet()` is an *event code*; each of which includes an *event value* that describes its details.

```
# Template of the event-based debugging
EvInit("buggy program name and its arguments")

while event := EvGet(eventmask, valuemask) do {
  case event of {
    E_Line : { /* handle breakpoints, stepping, etc */ }
    E_Assign |
    E_Value : { /* Handle assignment watchpoints */ }
    E_Deref : { /* Handle read watchpoints */ }
    E_Spos |
    E_Snew : { /* Handle string scanning environments */ }
    E_Error : { /* Handle a runtime error */ }
    E_Exit : { /* Handle buggy program normal exit */ }
  }
  /* Handle other debugging features such as tracing,
  profiling and internal and external debugging tools */
}
```

Figure 3. An example of AlamoDE debugging loop

`EvGet()` requests the next event by resuming the buggy program that is denoted by `&eventsourc`. A debugging tool can debug multiple buggy programs in one session. This can be used to perform advanced debugging techniques such as relative debugging [16] or delta

debugging [17]. Switching between different programs is accomplished by changing the value of `&eventsourc` before the next call to `EvGet()`. Furthermore, since different loaded programs are independent in their execution state, this architecture allows different debugging tools to be loaded under each other. It is possible for a debugging tool to debug another tool that is debugging the buggy program, or for multiple debugging tools to simultaneously debug the same buggy program during the same session and same run, see Figure 2.

### 3.4. Execution State Inspection/Modification

AlamoDE provides facilities to inspect the execution stack, check a variable state, and acquire information about the source code of the buggy program. Furthermore, it provides the ability to control and change the state of the buggy program by assigning to variables and redirecting procedures and functions.

**3.4.1. Variable State.** A variable is either global, or local including static and parameter variables. A local variable value can be obtained using the built-in function `variable(name, &eventsourc, level)`, which returns the current value of the variable `name` in the frame number `level` of the buggy program's call stack. If `name` is a global variable or a keyword, the same function is used without the `level` parameter (i.e. `variable(name,&eventsourc)`).

The `variable()` function is also used to assist a debugging tool in assigning to variables in the buggy program space. This mechanism introduces a potential safety problem if a context switch to the buggy program occurs between the time the variable reference is obtained and the time the assignment is complete. This problem is called *inter-program variable safety*, and it is solved by implementing a trapped variable technique, see Section 4.2.

**3.4.2. Stack Frames.** Activation records (frames) on the stack are distinguished by a positive integer called *level*; the most recent stack frame is at level zero, whereas the highest level value is for the activation record of procedure `main()`. The `proc()` built-in function was extended for AlamoDE to allow the debugging tool to identify which procedure is currently active on a specific stack level. For example, `proc(&eventsourc,7)` returns a pointer to the procedure/method, which lives on the seventh outer most level of the buggy program's call stack. The depth of the call stack can be checked using the keyword `&level`. The keyword `("&level",&eventsourc)` returns the number of frames currently on the buggy program's interpreter stack.

Furthermore, the Unicon language allows programmers to replace a procedure with another procedure during the execution. This feature is very useful for some debugging tools. For example, if the buggy program contains two versions of a sorting algorithm, in different procedures, the debugger can replace one by the other on the fly during the execution.

The procedure/method pointer obtained by the `proc()` function allows a debugging tool to place a call to that procedure as an *inter-program procedure call*. This mechanism is very useful for interactive source-level debuggers. For example, the buggy program may contain a procedure that prints the elements of a linked list, which is being debugged by the user. The debugger can place a call to that procedure, from any point during the debugging session, without modifying the buggy program source code. Moreover, a source-level debugger may incorporate general service procedures that can be plugged in to the buggy program source code on the fly during the debugging session. For example, the following assignment

```
variable("sort1",&eventsourc):=proc("qsort",&current)
```

replaces the buggy program's procedure `sort1()` with the debugger service procedure `qsort()`. Of course, the two procedures' formal parameters must be compatible in their order and type.

**3.4.3. Executed Source Code.** Unicon's executable bytecode contains information about the linked source files including any used library modules. For AlamoDE, a class library was developed to analyze the bytecode and generate a list of its source file names, and their static source code properties such as packages, classes, global variables, and user defined functions. Another class library maintains a list of all source files in use. Those library classes provide a debugging tool with the buggy program's source code static information.

Furthermore, the debugging tool can inspect the currently executed source code. Using functions such as `keyword()` as discussed in Section 3.2, and by monitoring runtime events such as `E_Line`, and `E_Syntax` that report the currently executed line number, and source code syntax construct respectively.

### 3.5. Advanced Debugging Support

AlamoDE provides underlying infrastructure for automatic debugging, dynamic analysis, profiling, and visualization. It utilizes many kinds of execution events that cover a wide range of execution behaviors, and gives the user the opportunity to try new debugging techniques. AlamoDE puts execution events in the hands of programmers, who can use events, event sequences, and event patterns to write their own automated debugging and dynamic analysis tools.

#### 3.5.1. Loading Secondary Debugging tools on the Fly.

AlamoDE allows advanced debugging techniques to coexist along with a classical debugger. It provides an on the fly extension mechanism. The `load()` function allows a debugger to incorporate external (secondary) standalone debugging tools under its control. The debugger coordinates and plays the role of a central server for other

```
$include "evdefs.icn"
link evint

class Example ( eventMask, gc, lastStr, lastBlk, collectedStr,
                collectedBlk, avgStr, avgBlk )

method handle_E_Collect()
  local Storage := []
  gc += 1
  every put(Storage, keyword("storage", Monitored))
  lastStr := Storage[2]; lastBlk := Storage[3]
end
method handle_E_EndCollect()
  local Storage := []
  every put(Storage, keyword("storage", Monitored))
  collectedStr += lastStr - Storage[2]
  collectedBlk += lastBlk - Storage[3]
end
method analyze_data()
  if gc > 0 then return 0
  avgStr := collectedStr / gc; avgBlk := collectedBlk / gc
end
method write_data()
  write("# Garbage Collections : ", gc)
  write(" Collected Strings : ", collectedStr, " Avg : ", avgStr)
  write(" Collected Blocks : ", collectedBlk, " Avg: ", avgBlk)
end
initially()
  eventMask := cset(E_Collect || E_EndCollect)
  gc := 0; collectedStr := collectedBlk := 0.0
end
procedure main(arg)
  EvInit(arg)
  obj := Example()

  while event := EvGet(obj.eventMask) do{
    case event of {
      E_Collect: { obj.handle_E_Collect() }
      E_EndCollect: { obj.handle_E_EndCollect() }
    }
    obj.analyze_data()
    obj.write_data()
  }
end
```

Figure 4. An AlamoDE debugging agent

debugging tools. The debugger and its loaded tools work synchronously on the same buggy program. For example, standalone debugging tools can be loaded on the fly during a source-level debugging session; without previous initialization. Loaded tools receive execution events from the host debugger, which multiplexes event codes and values using the function `EvSend()`, see Figure 2. Events are sent based on each tool event mask. An event is sent only to the tools that request it in their event mask.

For example, the code in Figure 4 shows a toy example of an AlamoDE-based debugging tool. It captures the number of garbage collections that happened during the execution of the buggy program, and finds the total and average of collected data from the string and block regions. This provides a rough measure whether the buggy program is mostly doing string processing or not. This example program can be used as standalone tool, or loaded into another debugging tool on the fly without any source code modification at all.

## 4. Implementation

This section provides an overview of the implementation of the most important underlying extensions. Some of the extensions are general additions to the Unicon virtual machine and its runtime system, while the rest are extensions to the Alamo monitoring framework.

### 4.1. Virtual Machine Instrumentation

Event-based debugging support needs instrumentation, which can be inserted into the source code, the bytecode, or implicit in the virtual machine itself. Implicit instrumentation is attractive to the end user, who needs a simple mechanism of getting events. However, instrumentation is always associated with overhead in space and processing time. Unicon has a small virtual machine (about 700KB with the instrumentation). A top priority for Unicon's implicit instrumentation is the processing time, which should not be affected for any unmonitored execution.

Originally, Alamo was an optional extension to the Icon virtual machine, because Alamo's instrumentation imposed a cost even when monitoring was not being performed. Alamo was integrated in the Unicon language with no measurable cost (other than code size) in the production virtual machine. This integration allows the debugger to run on the virtual machine synchronously along with the buggy program. The debugger and the buggy program run in two different co-expressions and the buggy program is the only one affected by the instrumentation.

AlamoDE maintains two versions of 30 runtime functions in the binary executable VM that contain instrumentation. One version is uninstrumented and used in any unmonitored execution; the other version is instrumented and used when a program is monitored. Furthermore, not all of the instrumented functions are used when the program is under monitoring; a dynamic binding associates the instrumented or uninstrumented function with the current execution state based on the current event mask, which is specified by the debugger. A table maps event codes into their instrumented functions. Whenever an event is added to the monitored events (event mask), the related instrumented function is used. If an event is removed from the event mask, the original uninstrumented version of the function is restored.

Inside the Unicon virtual machine source code, the name of the instrumented function uses the suffix “\_1”, whereas the name of the uninstrumented version of the same function uses the suffix “\_0”. Functions that contain instrumentation use macros to maintain one copy of the source code, which simplifies the maintenance effort.

Using this method of dynamic binding, the instrumentation imposes no cost on the execution time of the virtual machine until the program is debugged or monitored, and the only instrumented functions used are the ones relevant to the currently monitored events, which

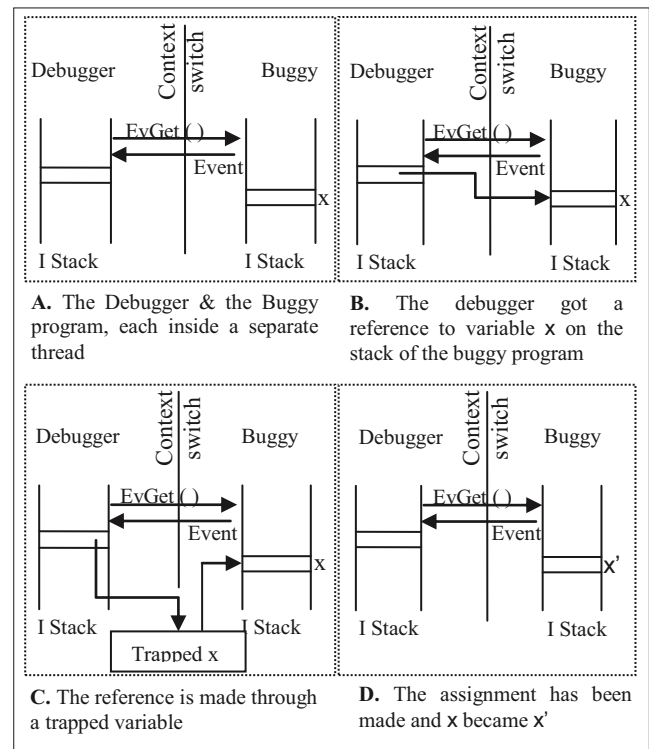


Figure 5. Trapped variable implementation

are specified by the event mask and customized by the debugging tool and the programmer.

### 4.2. Inter-Program Variable Safety

In order for a debugging tool to be able to change the value of a variable inside a buggy program, the tool must have access to the state of the buggy program. However, the debugging tool and its buggy programs are loaded into different co-expressions inside the Unicon's virtual machine.

Giving a co-expression the ability to access and change another co-expression state is critical. It is possible for one of the co-expressions to obtain a reference for a variable that is either in the stack, in the static data section, or in the heap of the other co-expression. However, while the first co-expression is trying to change a variable in the second co-expression, a context switch may allow control to be transferred to the second co-expression. A memory violation might occur if the second co-expression executes further while the first co-expression has a reference to a local variable; a reference to a variable that lives on the stack might become invalid. For example this can happen if the procedure returned and its activation record is popped off the stack. Since co-expressions are synchronous this is admittedly an unlikely occurrence that would only be caused by a deliberate adversary.

The implemented solution is a *trapped variable* technique [4]. Whenever one co-expression obtains a reference to the state of another co-expression, a trapped

variable block is allocated and the reference is done through that trapped variable, see Figure 5. The first co-expression contains a reference to a trapped variable block, which references the actual variable in the second co-expression. This new block holds information about the current number of context switches between the two co-expressions. This number is compared to the very recent one just before writing to that variable. If there is any difference between the number of context switches when the reference was obtained and when the reference is written, then this technique invalidates the assignment and gives the user a runtime error. This can only happen if a context switch occurs in the middle of an assignment to a *monitored trapped variable*. This new technique produces a runtime error where a monitor deliberately invokes the subject program in this way. This critical section can occur inside an Alamo monitor in unlikely scenarios such as:

```
1(variable("x", &eventsourc, 1), EvGet()) := 5
```

It is possible this way to call `EvGet()` and transfer control to the buggy program between the variable reference and its assignment, but it is not easy. Not surprisingly, the code for a normal debugger does not do any such thing. The safety feature was added to the language to extend the `variable()` function to produce references to local variables while a program is paused.

### 4.3. Syntax Instrumentation

Unicon's bytecode executes as a sequence of virtual machine instructions. Like most binary code formats, the bytecode formerly contained no information about the actual syntax of the source code. However, some automatic debugging facilities need information about the syntax of the running program. For example, an automated debugging technique that locates frequently failed loops needs to know when the execution of the buggy program enters and leaves a loop and what type of loop it is; such as a *while* loop.

The solution is to add a new pseudo instruction that is managed by the translator and the linker. The new `Op_Synt` syntax pseudo instruction extends the "line#/column#" table with information about the syntax. It is a reasonable solution because the only cost is a small increase in the size of the table. The cost of retrieving the syntax information from the table is paid for only when a program is monitored and that information is needed.

The "line#/column#" table was transformed into a "line#/column#/syntax" table without altering its design, see Figure 6. The table entry is a 32-bit integer; the most 16 significant bits were for the column number and the least 16 significant bits were for the line number. The maximum possible line/column number is 65535, which is more than is needed for a column number. We changed the column number bits to be the 11 most significant bits, and the remaining 5 bits are used for syntax information. The new

design changed the maximum possible column number to 2048, which is still more than enough for a column number. The newly added pseudo instruction only appears in the object files and is used by the linker while generating the bytecode. The new 5-bit syntax code can hold up to 32 different syntax indicators. AlamoDE presents syntax information as a new selectable event code `E_Syntax` and its related event value is the syntax code, see Table 1. A library routine was written to translate the syntax code into its symbolic name.

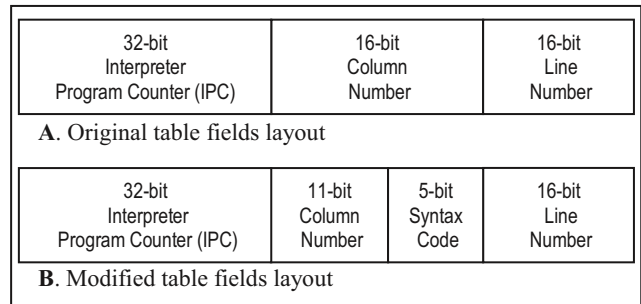


Figure 6. Unicon's line/syntax/column table

Table 1. Syntax events

Syntax	String Code	Integer Code
unidentified syntax	"any"	0
entering case expression	"case"	1
exiting case expression	"endcase"	2
entering if expression	"if"	3
exiting if expression	"endif"	4
entering if/else expression	"ifelse"	5
exiting if/else expression	"endifelse"	6
entering while loop	"while"	7
exiting while loop	"endwhile"	8
entering every loop	"every"	9
exiting every loop	"endevery"	10
entering until loop	"until"	11
exiting until loop	"enduntil"	12
entering repeat loop	"repeat"	13
exiting repeat loop	"endrepeat"	14
entering suspend loop	"suspend"	15
exiting suspend loop	"endsuspend"	16

## 5. Evaluation

An AlamoDE-based source-level debugger must use different approaches to implement features found in standard source-level debuggers, and faces potential performance challenges. In compensation, this type of implementation greatly simplifies the process of experimenting with new debugging techniques that probably would not be undertaken if the implementation was limited to the low-level approaches found in other debuggers. The AlamoDE debugging framework provides

high level facilities to customize and reduce the amount of monitored events and context switches.

AlamoDE was used to build an extensible source-level debugger called UDB [18], which integrates new automatic detection techniques that can be found in trace-based debuggers such as ODB [11,14]. One measurement of the effectiveness of the AlamoDE is that UDB's source code is less than 8K lines of source code. Furthermore, UDB provides two types of extension agents, externals (secondary debugging tools) and internals, supported by IDEA. IDEA is the Idaho Debugging Extension Architecture, a multi-agent debugging architecture built on top of AlamoDE and used by UDB. Each extension agent is a task-oriented program execution monitor. Debugging agents are standalone tools, which can be loaded on the fly during a UDB debugging session, or incorporated into its debugging core as permanent debugging features. Moreover, UDB's debugging agents can be enabled and disabled from any point during the debugging session and the user can be selective about which suite of agents to use.

Under UDB, eight different debugging agents were loaded and tested as external agents, and then adopted to become part of the UDB's library of internal agents. The slowdown imposed by the external agents was at most 3 times slower than the standalone agent mode, and the slowdown imposed by the migrated internal agents, was at most 2 times slower than the standalone agents. This suite of monitoring agents imposes at most 20 times slowdown on the execution of the buggy program over an uninstrumented execution mode, but in the general case, the slowdown depends on the algorithms used by the dynamic analysis technique implemented by the debugging agent. To place this in perspective, a debugger such as valgrind [15] imposes a 20 to 50 times slowdown, and it does not provide the interactive debugging environment that AlamoDE-based debugging tools provide, where the user can be selective about which and where to enable/disable agents from within a breakpoint based debugging session.

## 6. Related Work

Programming languages vary widely in their debugging support. Python provides debugging techniques through a module (PDB) [14], which maintains the classical debugging techniques such as breakpoints, stepping and continuing. It also supports post-mortem debugging and it can be called under program control. However, since PDB is a module, it must be imported into the Python program in order to be used. PDB was not designed with automatic debugging or extension particularly in mind, but the interpretive nature and high level of Python make it a good candidate for research experimentation. PDB's module architecture suggests that PDB perturbs application behavior such as garbage collection due to a shared heap.

The Smalltalk system includes very important tools such as a *browser*, *workspace*, *debugger*, and *inspector*. These tools provide a complete development and testing

environment system that assist in the *edit-compile-link-run-debug* cycle. All Smalltalk objects understand special messages such as `doesNotUnderstand` and `inspect`. The `doesNotUnderstand` message is produced automatically by Smalltalk runtime system as a result of a runtime error. This message causes the Smalltalk system to provide the user with an *error notification*, which asks the user if he/she is interested in a debugging session. During a debugging session, the programmer is able to modify the program while it is running. In general, Smalltalk runtime errors cause the execution thread to be suspended. In some cases the runtime error can be recoverable. The user may fix the error and continue in the execution. This simplifies the process of reproducing the bug. In fact, it is fairly common for programmers to struggle with this process. In contrast, the `inspect` message is produced and sent intentionally by the programmer; it allows a user to inspect an object through the inspector window [10].

SmallTalk's debugger has several similarities and important differences compared with UDB. The most important similarity is that both use a thread model of execution, which provides relatively good, high performance access to program state. Another similarity is that most of the debugger is written in the same language as the program that is being debugged. Compared with UDB, IDEA, and AlamoDE, SmallTalk's debugger is less separate from the program being debugged, and relies more on manual instrumentation via subclassing and overriding methods to generate events for dynamic analysis.

A debugging architecture such JPDA, with its latest lowest level JVM TI [5], provides an event-based debugging infrastructure and enough events for conventional debugging, profiling, and visualization. JVM TI supports about thirty five kinds of events, whereas AlamoDE incorporates more than one hundred kinds of events. Unicon programmers use events, event-sequences, and event-patterns to write their own debugging agents that detect specific execution behaviors—which some of they are suspicious behaviors while others are defined bugs. Furthermore, JVM TI is based on the inter-process communication for less intrusive on the buggy application. Unicon debugging support features the in-process communication with no intrusion on the buggy application.

Both AlamoDE and JVM TI, provide techniques to inspect the state and to control the buggy program running in the VM. JVM TI agents must be loaded and initialized at the start of the JVM, whereas different AlamoDE-based standalone debugging tools can work in concert with each other during a debugging session, or be incorporated into the debugger source code as permanent extensions, with little or no source code modification.

## 7. Future Work

Unicon's classical debugging support for features, such as breakpoints and watchpoints, is provided through monitored events and event filtering. Even though these

techniques perform well during debugging, improving their performance can be achieved by further implementation of common techniques such as *trapped virtual machine instruction* for breakpoints, and *trapped variable* for watchpoints. Moreover, AlamoDE's support for multiple simultaneous debugging tools can benefit from extending Unicon support with real concurrency, where different co-expressions can be off loaded onto different core processors.

AlamoDE's performance can be improved further by reducing the number of context switches. At present the monitor coordinator plays the role of a central server in a star network. A ring-based architecture where each monitor forwards events to another monitor instead of having a central coordinator would reduce context switches by 50%. Another possible architecture is a broadcasting mechanism where the buggy program broadcasts events to all secondary debugging tools. Furthermore, extending AlamoDE for other languages can benefit from existing instrumentation frameworks such as ASM for Java and PIN and ATOM for C and C++ programs.

## 8. Conclusion

Unicon's debugging support provides programmers with high level built-in functions and execution events that allow them to go beyond the standard debugging techniques. AlamoDE is designed to simplify automatic debugging, and dynamic analysis techniques. It allows various analyses to be used as standalone tools, dynamically loaded into a debugging coordinator with no source code alteration, or permanently incorporated into the debugging tool with minimal modifications. Programmers can utilize execution events, event patterns, and event sequences to capture specific behaviors. Some may be considered suspicious behaviors while others are classified as bugs.

The implementation of UDB proves that the AlamoDE framework is powerful enough to reduce the development cost of source-level debuggers and simplify its maintenance and extension. Previous event-based source-level debuggers, such as Dalek [12,13] identified performance obstacles. With this approach, AlamoDE provides usable debugging support proved in the implementation of UDB and its multi-agent debugging extension architecture.

## 9. Acknowledgment

This research was supported in part by an appointment to the National Library of Medicine Research Participation Program. This program is administered by the Oak Ridge Institute for Science and Education for the National Library of Medicine.

## 10. References

- [1] Auguston, M., Jeffery, C., and Underwood, S. 2002. A Framework for Automatic Debugging. In *Proceedings of the 17th IEEE International Conference on Automated Software Engineering* (September 23 - 27, 2002). Automated Software Engineering. IEEE Computer Society, Washington, DC.
- [2] Auguston, M., Jeffery, C., and Underwood. 2003. A Monitoring Language for Run Time and Post-Mortem Behavior Analysis and Visualization. In *proceedings of the Fifth International Workshop on Automated Debugging (AADEBUG 2003)*, September 2003, Ghent.
- [3] Griswold, R. E., and Griswold, M. T. 1997. *The Icon Programming Language*. Peer-to-Peer Communications, Inc., San Jose, California
- [4] Hanson, D. R, Variable Associations in SNOBOL4, *Software-Practice and Experience*, VOL. 6, 245-254 (1976)
- [5] Java Platform Debugging Architecture, <http://java.sun.com/javase/6/docs/technotes/guides/jpda>
- [6] Jeffery, C. L., 1999. *Program Monitoring and Visualization: an Exploratory Approach*, Springer New York
- [7] Jeffery, C. L., Mohamed, S., Pereda, R., and Parlett, R. 2004. *Programming with Unicon*. <http://unicon.org/book/ub.pdf>
- [8] Jeffery, C., Zhou, W., Templer, K., and Brazell, M. 1998. A lightweight architecture for program execution monitoring. *SIGPLAN Not.* 33, 7 (Jul. 1998), 67-74
- [9] K. Templer and C. Jeffery, "A Configurable Automatic Instrumentation Tool for ANSI C," Proc. 13th IEEE Int'l Conf. Automated Software Eng., pp. 249-258, 1998
- [10] LaLonde, W. R. and Pugh, J. R. 1990 *Inside Smalltalk: Vol. 1*. Prentice-Hall, Inc.
- [11] Lewis, B. *Debugging Backward in Time*. Proceedings of the Fifth International Workshop on Automated Debugging. AADEBUG 2003, September 2003, Ghent
- [12] Olsson, R. A., Crawford, R. H., and Ho, W. W. 1991. A dataflow approach to event-based debugging. *Software Practice & Experience*. 21:2 (Feb. 1991), 209-229
- [13] Olsson, R. A., Crawford, R. H., Ho, W. W., and Wee, C. E. 1991. Sequential Debugging at a High Level of Abstraction. *IEEE Software* 8:3 (May. 1991), 27-36
- [14] Pothier, G., Tanter, É., and Piquer, J. 2007. Scalable omniscient debugging. In *Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object Oriented Programming Systems and Applications* (Montreal, Quebec, Canada, October 21 - 25, 2007). OOPSLA '07. ACM, New York, NY, 535-552
- [15] Rossum, G. Python Library Reference: The Python Debugger. Release 2.5, 19th September, 2006. <http://www.python.org/doc/current/lib>
- [16] Searle, A.; Gough, J.; Abramson, D., "Automating relative debugging," *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, vol., no., pp. 356-359, 6-10 Oct. 2003.
- [17] Zeller, A.; Hildebrandt, R. "Simplifying and Isolating Failure-Inducing Input," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 183-200, February, 2002.
- [18] Ziad Al-Sharif. *Debugging with UDB, User Guide and Reference Manual*. Technical Report #10



# Pie Tree Visualization

Mireille Samia

Institute of Computer Science  
Heinrich-Heine-University Düsseldorf  
D-40225 Düsseldorf, Germany  
samia@cs.uni-duesseldorf.de

Michael Leuschel

Institute of Computer Science  
Heinrich-Heine-University Düsseldorf  
D-40225 Düsseldorf, Germany  
leuschel@cs.uni-duesseldorf.de

## Abstract

*Visualizing graphs with a large number of edges and vertices can be cumbersome and ineffective. This is due to the presence of countless overlapping arrows, which makes a graph unclear and hard to understand and interpret by a human. The aim of this paper is to try to address this problem using a new concept of data visualization, namely pie tree visualization. We illustrate this technique on the module architecture of a real-life development from the project Deploy. We first describe pie tree visualization, and then, present its advantages.<sup>1</sup>*

## 1. Introduction

Visualization is the presentation of information in a graphical format. It intends to provide the user with a qualitative understanding of this information. A comprehensive visualization should be:

- accurate: the presentation of the data is sufficient for its correct evaluation.
- effective: the information is easy to interpret.
- efficient: the information is appropriately and clearly shown.
- aesthetical: the data visualization must not offend the user's senses. It should be visually appealing.

Visualizing huge information, such as the structure of a large software project, can make a graph very confusing and ineffective, even though it is accurate. A typical example of such a graph is Figure 1, which depicts the module architecture of a formal software development from the Deploy

<sup>1</sup>This research is partially supported by the EU funded FP7 project 214158: DEPLOY (Industrial deployment of advanced system engineering methods for high productivity and dependability).

project. It is based on a common technique of data visualization of graphs and large software architectures, which is provided by *dot*, part of the GraphViz package [1]. In Figure 1, the original information is accurate; i.e., it remains unchanged in the visualization. However, due to countless overlapping arrows the graph becomes unreadable. Consequently, the user sees the information, but cannot understand the information's contents. Even a zooming into a portion of the graph of Figure 1 does not solve the problem (cf. Figure 2). First of all, the overall structure of the graph is lost. Moreover, Figure 2 shows clearly the overlapping arrows. More specifically, the nodes of the graph, depicting different modules, are connected together by unstructured arrows, which prevent to recognize the connections. Hence, a user is not able to see and to understand how the information of the graph is connected.

To prevent this problem, we propose a new data visualization, namely pie tree visualization. Pie tree visualization aims at presenting different components in a clear and visually appealing manner. More specifically, a node of a source (such as a module) is connected to other nodes using pie charts. Every slice of a pie chart represents a module. Every module has a name, a color and a number. By representing every module by a number, pie tree visualization considers the visual need of a color-blind user. Other advantages of pie tree visualization are that the number of connections of every module is clearly shown, as well as its characteristics.

In this paper, section 2 provides related work. In section 3, we present pie tree visualization by applying an instance to it, and discuss its advantages. Section 4 concludes this paper, and points out future work.

## 2. Related work

Graph visualization is defined as “the problem of constructing geometric representations of graphs, networks, and related combinatorial structures” [2]. There are several techniques for visualizing graphs or hierarchies, such as the classical hierarchical view [6]. Another technique is

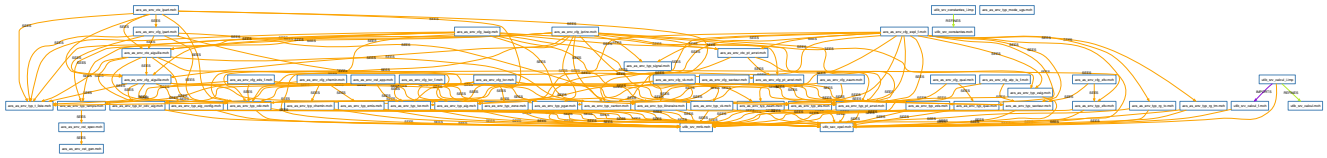


Figure 1. An accurate, but largely ineffective graph visualization.

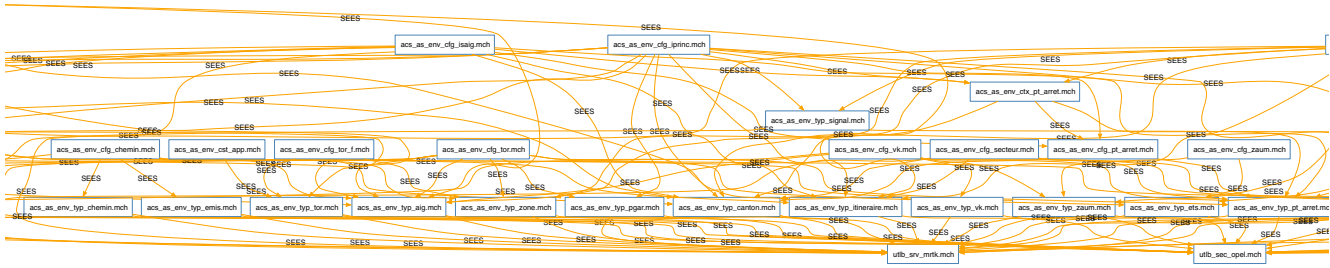


Figure 2. Portion of Figure 1, in a zoomed-in view, shows the countless overlapping arrows.

radial view, where the children of a sub-tree are positioned in circular wedges [3]. An extension to this technique is MoireGraphs for visualizing graphs [5]. Another work is fisheye views [4], which proposes a presentation strategy for hierarchical data structure. In this technique, an area of interest is enlarged, while other portions of the graph are shown with less detail. Another visualization technique is RINGS, a circular view in which all nodes and their children are placed in circle [8]. Another example is the visualization technique provided by *dot* [1]. The traditional techniques are accurate. However, as the amount of information to be visualized largely increases, they often become inappropriate. They are more concerned with the accurate visualization of data, and often do not consider the efficient visual presentation of information, which makes the data difficult to understand. They do not meet the visual need of a color-blind person. The aim of pie tree visualization is accuracy, but also comprehension. The user is able to find and access the needed information. Pie tree visualization is flexible and adaptable. It can adjust to serve multiple needs, such as the visual requirements of a color-blind user.

### 3. Pie tree visualization

In this section, we describe the new data visualization [7], namely pie tree visualization, by applying it to an industrial software architecture from the project Deploy. Then, we discuss its advantages.

#### 3.1. Description of pie tree visualization

Pie tree visualization consists of three main entities: the nodes of a source (such as modules), their connection(s),

and their corresponding legend.

The main goals of pie tree visualization are to represent *clearly* every module and its connection(s), to understand a module's characteristics, and to be user-friendly for color blind people.

Figure 8 illustrates pie tree visualization. It presents pie tree visualization, in a rotated view. It shows how the modules and the connections of the large software architecture of Figure 1 are represented in pie tree visualization. It depicts every module, as well as its characteristics.

**The legend** In Figure 3, every module is presented by a name, a number and a color. For instance, the module named “acs\_as\_env\_cst\_app” has the number 17, and a yellow tone color (*light yellow*).

1 acs_as_env_cfg_aiguille	16 acs_as_env_cfg_ziv	31 acs_as_env_pty_ligneaire	46 acs_as_env_pty_zds
2 acs_as_env_cfg_atp_is_f	17 acs_as_env_cst_app	32 acs_as_env_pty_mode_ugs	47 acs_as_env_pty_ziv
3 acs_as_env_cfg_chemin	18 acs_as_env_cst_gen	33 acs_as_env_pty_pgar	48 acs_as_env_pty_zone
4 acs_as_env_cfg_expl_f	19 acs_as_env_cst_spec	34 acs_as_env_pty_pt_arret	49 utlb_sec_opel
5 acs_as_env_cfg_ipart	20 acs_as_env_ctx_aiguille	35 acs_as_env_pty_quai	50 utlb_srv_calcul
6 acs_as_env_cfg_ipinc	21 acs_as_env_ctx_ipart	36 acs_as_env_pty_rg_tc	51 utlb_srv_calcul_f
7 acs_as_env_cfg_isaig	22 acs_as_env_ctx_pt_arret	37 acs_as_env_pty_rg_tm	52 utlb_srv_calcul_j
8 acs_as_env_cfg_pt_arret	23 acs_as_env_pty_aig	38 acs_as_env_pty_secteur	53 utlb_srv_constantes
9 acs_as_env_cfg_quai	24 acs_as_env_pty_aig_config	39 acs_as_env_pty_signal	54 utlb_srv_constantes_i
10 acs_as_env_cfg_secteur	25 acs_as_env_pty_br_cdv_aig	40 acs_as_env_pty_t_liste	55 utlb_srv_mrk
11 acs_as_env_cfg_tor	26 acs_as_env_pty_canton	41 acs_as_env_pty_temps	mch
12 acs_as_env_cfg_tor_f	27 acs_as_env_pty_cdv	42 acs_as_env_pty_tor	imp
13 acs_as_env_cfg_vk	28 acs_as_env_pty_chemin	43 acs_as_env_pty_vk	SEES
14 acs_as_env_cfg_zaum	29 acs_as_env_pty_emis	44 acs_as_env_pty_zaig	IMPORTS
15 acs_as_env_cfg_zds_f	30 acs_as_env_pty_ets	45 acs_as_env_pty_zaum	REFINES

Figure 3. The legend of pie tree visualization.

We use a range of color tones to represent, for example, different types of modules. In Figure 3, the different types that a module can belong to are *cfg*, *cst*, *ctx*, *pty*, *sec* and *srv*. Every type of a module is given a different color. For instance, the *cst* modules, numbered from 17 to 19, have a range of *yellow* tones. By giving various types of modules different colors, we group the modules of similar types.

In the case of Figure 3, a module can be of different

kinds: either an abstract machine (.mch) or an implementation (.imp). An abstract machine, is illustrated visually as a full rectangle, and an implementation machine (.imp) as a dashed rectangle.

Using distinct types of rectangles helps to see quickly if a module is of type “machine” or “implementation”.

Every connection of two modules has a label. In Figure 3, the labels are SEES, IMPORTS and REFINES. Each label is represented by a different arrow, which helps to know quickly the corresponding label of a connection.

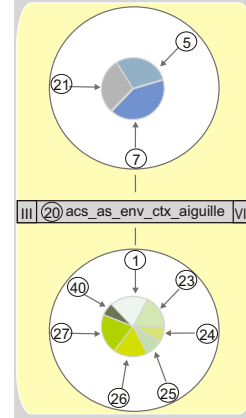
**A module** As stated previously, a machine is depicted by a full rectangle, and an implementation by a dashed rectangle. In every rectangle of a module, the number of incoming and outgoing connections are stated by a roman number or 0 (if no connection). The left side of the rectangle shows the number of the incoming connections of a module; i.e., the connections to a module. The right side of the rectangle shows the number of outgoing connections of a module; i.e., the connections from a module to different modules.

Figure 4 gives an example of a machine, named “acs\_as\_env\_ctx\_aiguille”, having the number 20 and the color *light grey*. The number of connections to the machine “acs\_as\_env\_ctx\_aiguille”; i.e. the incoming connections, is the roman number *III*. The number of connections from the machine “acs\_as\_env\_ctx\_aiguille” to other different modules; i.e. the outgoing connections, is the roman number *VII*. In Figure 5, an instance of an implementation, called “utlb\_srv\_calcul\_i”, numbered 52 and colored *light violet*, is illustrated. The number of incoming connections is 0. In other words, none of the modules is connected to the implementation “utlb\_srv\_calcul\_i”. The number of outgoing connections, which represent the connections from “utlb\_srv\_calcul\_i” to other modules, is *III*.

**The connections of a module** The connection(s) from or to a module is/are depicted by a pie. The upper part connected to a rectangle, which represents a module, shows the connections from other different modules to this module. The lower part displays the connections from a module (rectangle) to other different modules.

Every slice of a pie chart depicts a module. Its color is that of the corresponding module. Its number is also of the corresponding module, and is shown in a circle. If the outline of the circle is dashed, then the module is an implementation. Otherwise, the module is a machine. The size of a pie’s slice represents a characteristic of a module, such as the total number of occurrences of a module or the size of a module. The labels of the connection between two different modules is represented by different arrows, as stated previously.

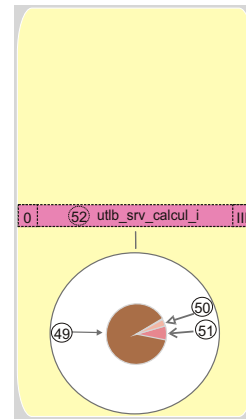
In Figure 4, for instance, the connections from other modules to the machine “acs\_as\_env\_ctx\_aiguille”, numbered 20, are the machines, represented by full circles numbered



**Figure 4. Instance of a machine.**

5, 7 and 21. Moreover, the connections from this machine to other different modules are the machines numbered 1, 23, 24, 25, 26, 27 and 40. The size of every pie slice represents the total number of occurrences of every module. All the labels of the connections between two different modules are depicted by a full arrow, which denotes SEES.

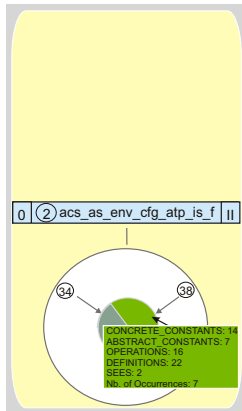
Another example is Figure 5, which shows the implementation “utlb\_srv\_calcul\_i” numbered 52, and having no connection from other modules to it; i.e. no incoming connection. The outgoing connections; i.e. the connections from the implementation “utlb\_srv\_calcul\_i” to other different modules, are to the machines 49, 50 and 51. The machine numbered 49 has the highest number of occurrences. The labels of the connections are shown by different arrows, which represent SEES, REFINES and IMPORTS.



**Figure 5. Instance of an implementation.**

**The summary of a module** A summary provides information about a module, such as, the total number of a module’s occurrences, the total number of a connection’s label, the total number of constants or of variables used in this

module, the total number of the definitions used in this module, the total number of a module's operations, and so on. The summary of a module is shown by a mouse click on the corresponding module, represented by a rectangle or a pie slice. Figure 6 shows the summary of machine 38, which has 14 concrete\_constants, 7 abstract\_constants, 16 operations, 22 definitions, 2 labels SEES, and its total number of occurrences is 7.



**Figure 6. Instance of a machine and its summary.**

### 3.2. Advantages of pie tree visualization

Pie tree visualization has several advantages, described below.

The user does not need to follow countless overlapping arrows. In pie tree visualization, every module is represented individually (cf. Figure 8). Consequently, even for large software architectures, pie tree visualization remains comprehensible. It improves the ease of interpretation of the information of every module, since in order to find the corresponding connections to and from a module, the user does not follow overlapping arrows, as in the visualization of graphs (cf. Figures 1 and 2).

The legend of pie tree visualization gives an overview of every module, such as its color, its number and its type. It also provides the total number of modules. In case of Figure 3, the total number of modules is 55. This helps to know how many modules are involved.

The summary of a module gives instantly an informative overview of a particular module.

The use of different arrows makes it easy to know the label of a connection. The labels are not overlapped, as in Figures 1 and 2.

The use of distinct outlines of circles and rectangles (full or dashed) gives the opportunity to promptly interpret the type of a module. A user is able to know visually and quickly, if a module is a machine or an implementation.

Pie tree visualization is user-friendly for color blind users. By representing a module by a number, a color blind user can directly identify which module is in question, without the need to identify the module's color. The text and the outlines of the shapes used, such as rectangles, arrows and circles, are colored in black tones. This facilitates to see *well* the information of a module.

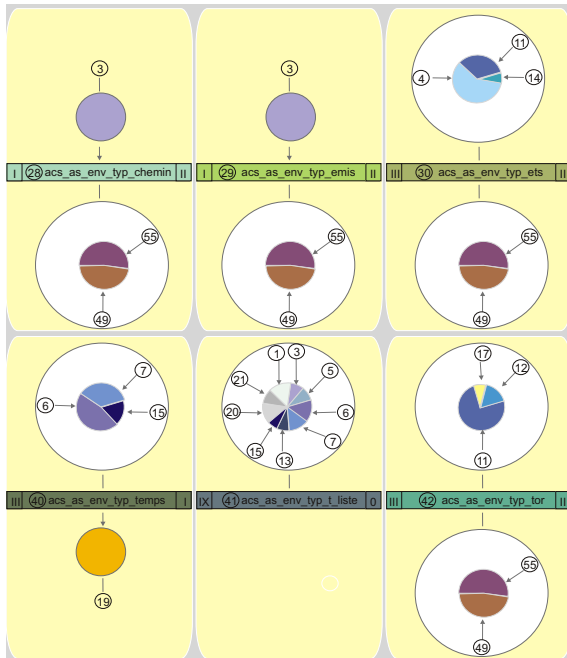
Giving the same type of modules a similar range of color tones facilitates to find out to which type a module belongs to. For instance, in Figure 3, the modules of type *ctx* can be directly recognized by their range of *grey* tones. Another way to know the type of a module is to check the name of the modules in the legend, which help to find out the range of numbers of the modules having the same type. For instance, the numbers from 20 to 22 have the same type *ctx*.

The size of every slice can be used to determine a specific characteristic of a module. In our case, in Figure 8, the size of a pie slice corresponds to the total number of occurrences of a module. By checking the size of a pie slice, we can easily and quickly visually estimate the corresponding characteristic of a module.

Pie tree visualization decreases the time of search for specific modules and decreases erroneous conclusions, since the user does not need to follow overlapping arrows, as in Figures 1 and 2. We can find a module easily and quickly by checking its number or its color. For instance, in Figure 8, the module “acs\_as\_env\_cfg\_aiguille” numbered 1 is connected to the machine “acs\_as\_env\_typ\_aig\_config” numbered 24, to the machine “acs\_as\_env\_typ\_br\_cdv\_aig” numbered 25, and so on.

Pie tree visualization outperforms the visualization provided by *dot*, in Figures 1 and 2 for certain complicated comparisons. The comparison of machines and implementations is easier with pie tree visualization than with graphs. If we need to compare which modules have the same connections, we do not need to follow arrows as in graphs. A user needs to compare the color of a module or its number which may occur in the other different modules. Then, the user can quickly identify, for instance, which modules have the same number of connections or which modules have similar connections. As stated previously, the upper connections to a module are the incoming connections of a module; i.e., the connections from other different modules to this module. The lower connections are the outgoing connections of a module; i.e. the connections from this module to other different modules. For instance, in Figure 7, machine “acs\_as\_env\_typ\_chemin” has the same incoming connection numbered 3, as machine “acs\_as\_env\_typ\_emis” and machine “acs\_as\_env\_typ\_t\_liste”. The machine “acs\_as\_env\_typ\_temps” has no common *lower* or outgoing connection with the machines having the numbers 28, 29, 30, 41 and 42. The machine numbered 41 has the highest number of incoming connections, which is the roman num-

ber IX. Note that by checking the number of every module in the legend, the name of every module is also recognized.



**Figure 7. Instances of modules having the same or different connections.**

A comparison between all the modules can be done by referring to Figure 8. A user is able to see and compare how all the modules are connected. He/she can easily see, and also at a glance, which modules have no outgoing or no incoming connections. For instance, it is also possible to find which modules have the same number of outgoing or incoming connections. After that, a user can compare which connections of the modules are similar, if any.

Providing a user with a qualitative visual presentation of the information is very important. This helps a user to better understand how the information interacts with each other. It also augments his/her comprehension of the data, which supports the interpretation of the information. By using pie tree visualization, the information is appropriately shown. We believe that a user can easily and quickly gain information, which helps and supports his/her understanding of this information.

#### 4. Conclusion and future work

Every visualization has its advantages and its disadvantages. The visualization of graphs provided, for instance, by *dot* performs well on a small amount of data. However, as the amount of information to be visualized becomes larger, its output tends to become confusing and ineffective, because of countless overlapping arrows. Consequently, the

connections between the modules may be hard to recognize, which makes the information difficult to understand by a human. Pie tree visualization is not always useful for a small amount of information. However, as the amount of data increases, we believe that the use of pie tree makes sense. The user does not need to follow countless arrows in order to find the required information and understand it. Pie tree visualization connects different nodes of a source (such as modules of a large software architecture) in a structured and visually appealing manner. A module is connected to other modules using pie charts, where every pie slice depicts a module. The connections to and from a module are clearly represented. Pie tree visualization is adequate for complicated comparisons. Due to the use of numbers, text and a correct combination of different modules' outlines, pie tree visualization may be user-friendly for color-blind persons. We believe that pie tree visualization is visually informative and comprehensive.

A future work is to add more features to pie visualization, such as improving the comparison between modules by, for instance, giving a more detailed summary of every module.

#### Acknowledgements

We would like to thank Jérôme Falampin for providing us with the industrial example of the large software architecture from the project Deploy, which is shown in Figure 1.

#### References

- [1] AT&T Labs-Research. Graphviz - open source graph drawing software. Obtainable at <http://www.research.att.com/sw/tools/graphviz/>.
- [2] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, New Jersey, 1999.
- [3] P. Eades. Drawing free trees. *Bulleting of the Institute of Combinatorics and its Applications*, pages 10–92, 1992.
- [4] G. W. Furnas. Generalized fisheye views. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'86)*, pages 16–23, New York, 1986. ACM Press.
- [5] T. J. Jankun-Kelly and K.-L. Ma. Moiregraphs: Radial focus+context visualization and interaction for graphs with visual nodes. *IEEE Symposium on Information Visualization 2003*, pages 59–66, 2003.
- [6] E. M. Reingold and J. S. Tilford. Tidier drawing of trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, 1981.
- [7] M. Samia and M. Leuschel. Towards pie tree visualization of graphs and large software architectures. In *the 17th IEEE International Conference on Program Comprehension, ICPC 2009*. IEEE Computer Society, 2009. to appear.
- [8] S. T. Teoh and K.-L. Ma. Rings: A technique for visualizing large hierarchies. In *Graph Drawing 2002*, pages 268–275, Irvine, California, USA, 2002. Springer.

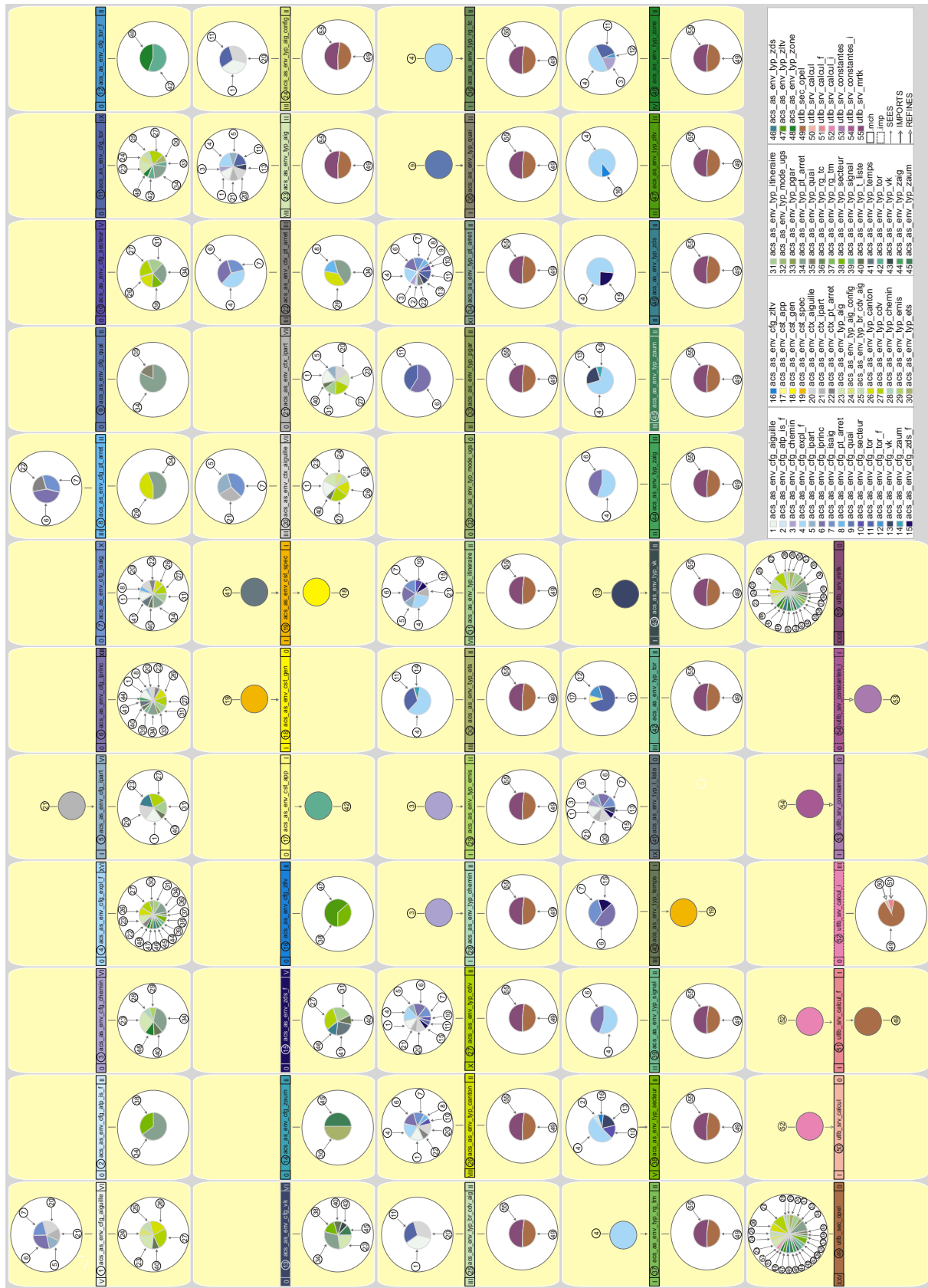


Figure 8. Pie tree visualization of the graph from Figure 1. (Rotated view).

# Formal Verification of Scalable NonZero Indicators

Shao Jie Zhang<sup>†</sup>, Yang Liu<sup>†</sup>, Jun Sun<sup>†</sup>, Jin Song Dong<sup>†</sup>, Wei Chen<sup>‡</sup> and Yanhong A. Liu<sup>\*</sup>

<sup>†</sup> National University of Singapore

shaojiezhong@nus.edu.sg, {liuyang,sunj,dongjs}@comp.nus.edu.sg

<sup>‡</sup> Microsoft Research Asia  
weic@microsoft.com

<sup>\*</sup> State University of New York at Stony Brook  
liu@cs.sunysb.edu

## Abstract

*Concurrent algorithms are notoriously difficult to design correctly, and high performance algorithms that make little or no use of locks even more so. In this paper, we describe a formal verification of a recent concurrent data structure Scalable NonZero Indicators. The algorithm supports incrementing, decrementing, and querying the shared counter in an efficient and linearizable way without blocking. The algorithm is highly non-trivial and it is challenging to prove the correctness. We have proved that the algorithm satisfies linearizability, by showing a trace refinement relation from the concrete implementation to its abstract specification. These models are specified in CSP and verified automatically using the model checking toolkit PAT.*

## 1 Introduction

Concurrent algorithms are notoriously difficult to design correctly, and high performance algorithms that make little or no use of locks even more so. The main correctness criterion of the concurrent algorithm design is linearizability [6]. Informally, a shared object is *linearizable* if each operation on the object can be understood as occurring instantaneously at some point, called *linearization point*, between its invocation and its response, and its behavior at that point is consistent with the specification for the corresponding sequential execution of the operation.

Formal verification of linearizability is challenging because the correctness often relies on the knowledge of linearization points, which is difficult or even impossible to identify. These proofs are too long and complicated to do (and check) reliably “by hand”. Hence, it is important to develop techniques for mechanically performing, or at least checking, such proofs.

In this paper, we present an approach to verify linearizability based on refinement relations between abstract spec-

ification and concrete implementation models of a concurrent algorithm. Both are specified using an event-based modeling language, which has formal semantics based on labeled transition systems. We have used this approach to formally verify a recent concurrent algorithm Scalable NonZero Indicators (SNZI) due to Ellen et al. [5], since the algorithm as a complex and useful implementation serves a good candidate for automatic verification. Our approach also builds on earlier work [8] in which we proved (and in some cases disproved and/or improved) a number of concurrent algorithms like nonblocking stacks, nonblocking queues, K-valued Registers and Mailbox problem. We have made considerable progress in understanding how to model algorithms including specifications and implementations to allow model checking to scale up and handle bigger cases. The complete model of SNZI algorithm is built inside a novel model checking tool, PAT [12] (<http://pat.comp.nus.edu.sg>).

The rest of the paper is structured as follows. Section 2 briefly introduces the SNZI algorithm. Section 3 gives the standard definition of linearizability. Section 4 shows how to express linearizability using refinement relations in general. Section 5 gives the SNZI model in our modeling language. Section 6 presents the verification and experimental results. Section 7 discusses related work and concludes.

## 2 The SNZI Algorithms

A SNZI object behaves similarly to traditional shared counter. It has one shared integer variable *surplus* and supports three operations: *Arrive* increments *surplus* by 1 when a process enters; *Depart* decrements *surplus* by 1 when the process leaves; the only difference from traditional counters is *Query* operation: it returns a boolean value indicating whether the value of *surplus* is greater than 0. We assume that each *Arrive* operation is always followed by a *Depart* operation for the same process. Therefore *surplus* is always greater or equal to 0. The pseudo code in Fig. 1 gives the

```

shared variable : Surplus : integer ; initially 0
bool Query() : return (Surplus > 0)
void Arrive() : Surplus ← Surplus + 1
void Depart() : Surplus ← Surplus - 1

```

**Figure 1. SNZI specification**

specification of a SNZI object.

In [5], the authors propose a rooted tree as the underlying data structure of the SNZI objects implementation. An operation on a child node may invoke operations on its parent. An important invariant is used to guarantee the correctness: the *surplus* of parent node is non-zero if and only if there exists at least one child whose *surplus* is non-zero. Thus, if the *surplus* of one node in the tree is non-zero, so does the root. A process begins *Arrive* operation on any node as long as the corresponding *Depart* will be invoked at the same node, and *Query* operation is directly invoked on the root. Every tree node has a counter  $X$  that is increased by *Arrive* and decreased by *Depart*. Since the operations on hierarchical nodes differ from those on root node, the algorithms are separated for hierarchical nodes and root node.

The code for hierarchical SNZI nodes is shown in Fig 2. An *Arrive* operation on a hierarchical node invokes *Arrive* operation on its parent node when increasing  $X$  from 0 to 1. Otherwise, it completes without invoking any operation. Moreover, a process which increases  $X$  from 0 to 1 should firstly set  $X$  by an intermediate value  $\frac{1}{2}$ . Any process which sees  $\frac{1}{2}$  must help that process to invoke *parent.Arrive* and try to change  $X$  to 1. If a process succeeds in invoking *parent.Arrive* but fails in setting  $X$  to 1, it will invoke a compensating *parent.Depart*.

Similarly, a *Depart* operation on a hierarchical node only invokes *Depart* on its parent node when decreasing  $X$  from 1 to 0. A version number is added to  $X$  to ensure that every change of  $X$  will be detected in both *Arrive* and *Depart* operations for hierarchical nodes as well as root node.

The code for root node is shown in Fig 3. In order to reduce frequent accesses to  $X$  by *Query*, the solution for the root node separates out an indicator bit  $I$  from  $X$ . Hence every process can finish *Query* only by reading the bit  $I$ . The authors model all accesses to  $I$  using *Read*, *Write*, *Load Linked* and *Store Conditional* primitives to tolerate spurious failures when external applications try to modify  $I$ .

$I$  is set to true after a 0 to 1 transition of  $X$ , and it is unset to false after a 1 to 0 transition of  $X$ . Furthermore, an announce bit  $a$  is added to  $X$  to indicate that  $I$  needs to be set. Similar to the intermediate value  $\frac{1}{2}$ , a process should set  $a$  during a 0 to 1 transition and clean it after setting  $I$  successfully. Any other process will also set  $I$  if it sees that  $a$  is set. Once the indicator is set, it can safely clear  $a$  to prevent unnecessary future writes to the indicator.

```

shared variables:
 $X = (c, v) : (\mathbb{N} \cup \{\frac{1}{2}\}, \mathbb{N})$ ; initially(0, 0)
parent: scalable indicator
Arrive
  succ ← false
  undoArr ← 0
  while( $\neg$ succ)
     $x \leftarrow$  Read(X)
    if  $x.c \geq 1$  then
      if CAS( $X, x, (x.c + 1, x.v)$ ) then
        succ ← true
    if  $x.c = 0$  then
      if CAS( $X, x, (\frac{1}{2}, x.v + 1)$ ) then
        succ ← true
         $x \leftarrow (\frac{1}{2}, x.v + 1)$ 
    if  $x.c = \frac{1}{2}$  then
      parent.Arrive
      if  $\neg$ CAS( $X, x, (1, x.v)$ ) then
        undoArr = undoArr + 1
  while(undoArr > 0) do
    parent.Depart
    undoArr = undoArr - 1
Depart
  while(true) do
     $x \leftarrow$  Read(X)
    if CAS( $X, x, (x.c - 1, x.v)$ ) then
      if  $x.c = 1$  then parent.Depart
    return

```

**Figure 2. Code for hierarchical SNZI node**

### 3 Linearizability

Linearizability [6] is a safety property of concurrent systems. It is formalized as follows.

In a shared memory model  $\mathcal{M}$ ,  $O = \{o_1, \dots, o_k\}$  denotes the set of  $k$  shared objects,  $P = \{p_1, \dots, p_n\}$  denotes the set of  $n$  processes accessing the objects. Shared objects support a set of *operations*, which are pairs of invocations and matching responses. Every shared object has a set of states that it could be in. A *sequential specification* of a (deterministic) shared object is a function that maps every pair of invocation and object state to a pair of response and a new object state.

The behavior of  $\mathcal{M}$  is defined as  $H$ , the set of all possible sequences of invocations and responses together with the initial states of the objects. A history  $\sigma \in H$  induces an irreflexive partial order  $<_\sigma$  on operations such that  $op_1 <_\sigma op_2$  if the response of  $op_1$  occurs in  $\sigma$  before the invocation of  $op_2$ . Operations in  $\sigma$  that are not related by  $<_\sigma$  are concurrent.  $\sigma$  is sequential iff  $<_\sigma$  is a strict total order. Let  $\sigma|_i$  be the projection of  $\sigma$  on process  $p_i$ , which is the subsequence of  $\sigma$  consisting of all invocations and re-



```

shared variables:
 $X = (c, a, v) : (\mathbb{N}, \text{boolean}, \mathbb{N})$ ; initially(0, false, 0)
 $I : \text{boolean}$ ; initially false
Arrive
  repeat
     $x \leftarrow \text{Read}(X)$ 
    if  $x.c = 0$  then  $x' \leftarrow (1, \text{true}, x.v + 1)$ 
    else  $x' \leftarrow (x.c + 1, x.a, x.v)$ 
  until  $\text{CAS}(X, x, x')$ 
  if  $x'.a$  then
     $\text{Write}(I, \text{true})$ 
     $\text{CAS}(X, x', (x'.c, \text{false}, x'.v))$ 
Depart
  repeat
1.    $x \leftarrow \text{Read}(X)$ 
2.   if  $\text{CAS}(X, x, (x.c - 1, \text{false}, x.v))$  then
3.     if  $x.c \geq 2$  then
4.       repeat
5.          $\text{LL}(I)$ 
6.         if  $\text{Read}(X).v \neq x.v$  then return
7.         if  $\text{SC}(I, \text{false})$  then return
Query
  return  $\text{Read}(I)$ 

```

**Figure 3. Code for SNZI root node**

sponses that are performed by  $p_i$ . Let  $\sigma|_{o_i}$  be the projection of  $\sigma$  on object  $o_i$ , which consists of all invocations and responses of operations that are performed on object  $o_i$ .

A sequential history  $\sigma$  is *legal* if it respects the semantics of the objects as expressed in their sequential specifications. More specifically, for each object  $o_i$ , if  $s_j$  is the state of  $o_i$  before the  $j$ -th operation  $op_j$  in  $\sigma|_{o_i}$ , then the invocation and response of  $op_j$  and the resulting new state  $s_{j+1}$  of  $o_i$  follow the sequential specification of  $o_i$ . Given a history  $\sigma$ , a *sequential permutation*  $\pi$  of  $\sigma$  is a sequential history in which the set of operations as well as the initial states of the objects are the same as in  $\sigma$ . The formal definition of linearizability is given as follows.

**Linearizability** There exists a sequential permutation  $\pi$  of  $\sigma$  such that 1) for each object  $o_i$ ,  $\pi|_{o_i}$  is a legal sequential history (i.e.  $\pi$  respects the sequential specification of the objects), and 2) if  $op_1 <_{\sigma} op_2$ , then  $op_1 <_{\pi} op_2$  (i.e.,  $\pi$  respects the real-time ordering of operations).

In every history  $\sigma$ , if we assign increasing time values to all invocations and responses, then every operation can be shrunk to a single time point between its invocation and response such that the operation appears to be completed instantaneously at this time point [3]. This time point for each operation is called its *linearization point*. Linearizability is defined in terms of the invocations and responses

of high-level operations, which are implemented by algorithms on concrete shared data structures in real programs. Therefore, the execution of high-level operations may have complicated interleaving of low-level actions. Linearizability of a concrete concurrent algorithm requires that, despite of complicated low-level interleaving, the history of high-level interface events still has a sequential permutation that respects both the real-time ordering among operations and the sequential specification of the objects. This idea is formally presented in Section 4 using refinement relations.

## 4 Verification via Refinement Checking

We model concurrent systems using a process algebra, whose behavior is described using a labeled transition system. Linearizability is then defined as a refinement relation from an implementation model to a specification model.

### 4.1 Modeling Language

We introduce the relevant subset of syntax of CSP (Communicating Sequential Processes) [7] extended with shared variables. We choose this language because of its rich set of operators for concurrent communications.

**Process** A process  $P$  is defined using the grammar:

$$P ::= \text{Stop} \mid \text{Skip} \mid e\{\text{program}\} \rightarrow P \mid P \setminus X \mid P_1; P_2 \mid P_1 \square P_2 \mid \text{if}(b) \{P_1\} \text{ else } \{P_2\} \mid P_1 \parallel P_2$$

where  $P, P_1, P_2$  are processes,  $e$  is a name representing an event with an optional sequential program *program*,  $X$  is a set of events, and  $b$  is a Boolean expression.

*Stop* is the process that communicates nothing, also called deadlock.  $\text{Skip} = \checkmark \rightarrow \text{Stop}$ , where  $\checkmark$  is the termination event. Event prefixing  $e \rightarrow P$  performs  $e$  and afterwards behaves as process  $P$ . If  $e$  is attached with a sequential program, the valuation of the shared variables is updated accordingly. For simplicity, assignments are restricted to update only shared variables. Process  $P \setminus X$  hides all occurrences of events in  $X$ . An event is invisible iff it is explicitly hidden by the hiding operator  $P \setminus X$ . Sequential composition,  $P_1; P_2$ , behaves as  $P_1$  until its termination and then behaves as  $P_2$ . External choice  $P_1 \square P_2$  is solved only by the occurrence of an visible event. Conditional choice  $\text{if}(b) \{P_1\} \text{ else } \{P_2\}$  behaves as  $P_1$  if the Boolean expression  $b$  evaluates to true, and behaves as  $P_2$  otherwise. Indexed interleaving  $P_1 \parallel P_2$  runs all processes independently except for communication through shared variables. Processes may be recursively defined, and may have parameters (see examples later). The formal syntax and semantics of our language is presented in [11].

To model nonblocking algorithms, our language provides strong support for synchronization primitives, such as *compare – and – swap* (CAS) and *load – linked* (LL)/*store – conditional* (SC), which are elaborated as follows.

**CAS**<sup>1</sup> The operational semantics of conditional choice requires that the condition evaluation and the first event to be executed of true/false branch be finished in one atomic step. Hence CAS primitive can be directly modeled using conditional choices.

```

/* The pseudo code of CAS semantics */
bool CAS(ref addr, val exp, val new) :
  atomically {
    if(*addr == exp) {*addr := new; }
    else { }
  }
/* The CSP representation of CAS */
if(*addr == old) {τ{*addr = new; } → Skip}
else {Skip}

```

**LL/SC**<sup>2</sup> In our model, a shared counter *counter* is added to indicate the timestamp when the content of a memory location *X* is modified and a counter flag is associated with each process. When LL is executed by one of the processes, the content of *X* is read and the value of *counter* is stored in the counter flag. If an external event updates *X* or the process executes an operation that may invalidate an atomic sequence (e.g., an exception), then *counter* is increased by 1. When the corresponding SC is executed, the counter flag is checked. If the flag is equal to *counter*, then SC will be successfully executed. Otherwise, nothing can be done.

```

/* flag[i] denotes the counter flag of process i */
LL(i) = τ{READ X; flags[i] = counter; } → Skip;
SC(i, v) = if(flags[i] == counter)
           {τ{X = v; counter++; } → Skip}
           else Skip;
Update(v) = τ{UPDATE X; counter++; } → Skip;

```

The semantics of a model is defined using a labeled transition system (LTS). Let  $\Sigma$  denote the set of all visible events and  $\tau$  denote the set of all invisible events. Let  $\Sigma^*$  be the set of finite traces. Let  $\Sigma_\tau$  be  $\Sigma \cup \tau$ . A LTS is a 3-tuple  $L = (S, init, T)$  where  $S$  is a set of states,  $init \in S$  is the initial state, and  $T \subseteq S \times \Sigma_\tau \times S$  is a labeled transition relation. Let  $s, s'$  be states in  $S$  and  $e \in \Sigma_\tau$ , we write  $s \xrightarrow{e} s'$  to denote  $(s, e, s') \in T$ . We write  $s \xrightarrow{e_1, e_2, \dots, e_n} s'$  iff there exists  $s_1, \dots, s_{n+1} \in S$  such that  $s_i \xrightarrow{e_i} s_{i+1}$  for all  $1 \leq i \leq n$ ,  $s_1 = s$  and  $s_{n+1} = s'$ . Let  $tr : \Sigma^*$

<sup>1</sup>CAS atomically compares the content of a memory location to an expected value, and if they are the same, the content of that memory location is assigned to the new given value.

<sup>2</sup>LL/SC are a pair of instructions. LL first reads the current content from a memory location *X*. A subsequent SC stores a new value to *X* only if no updates have happened in between LL and SC; otherwise, it fails.

be a sequence of visible events.  $s \xrightarrow{tr} s'$  iff there exists  $e_1, e_2, \dots, e_n \in \Sigma_\tau$  such that  $s \xrightarrow{e_1, e_2, \dots, e_n} s'$ . The set of traces of  $L$  is  $traces(L) = \{tr : \Sigma^* \mid \exists s' \in S, init \xrightarrow{tr} s'\}$ . In this paper, we consider only LTSs with a finite number of states. In particular, we bound the sizes of variable domains by constants, which also bounds the depths of recursions.

**Theorem 1** (Refinement). *Let  $L_{im} = (S_{im}, init_{im}, T_{im})$  be a LTS for an implementation. Let  $L_{sp} = (S_{sp}, init_{sp}, T_{sp})$  be a LTS for a specification.  $L_{im}$  refines  $L_{sp}$ , written as  $L_{im} \sqsupseteq_T L_{sp}$ , iff  $traces(L_{im}) \subseteq traces(L_{sp})$ .*

## 4.2 Linearizability

This section briefly shows how to create high-level linearizable specifications and how to use refinement relation to define linearizability of concurrent implementations.

We define the linearizable specification LTS  $L_{sp} = (S_{sp}, init_{sp}, T_{sp})$  for a shared object  $o$  in the following way. Every execution of an operation of  $o$  on a process includes three atomic steps: the invocation action, the linearization action, and the matching response action. The linearization action performs the computation based on the sequential specification of the object. All the invocation and response actions are visible events, while the linearization ones are invisible events. Their complete specification and transition rules in LTS is formally presented in [8]. We now consider a LTS  $L_{im} = (S_{im}, init_{im}, T_{im})$  that supposedly implements object  $o$ . Theorem 2 characterizes linearizability of the implementation through refinement relations.

**Theorem 2.** *Traces of  $L_{im}$  are linearizable iff  $L_{im} \sqsupseteq_T L_{sp}$ .*

The proof of theorem 2 is given in [8]. The theorem shows that to verify linearizability of an implementation, it is necessary and sufficient to show that the implementation LTS is a refinement of the specification LTS as we defined above. This provides the theoretical foundation of our verification of linearizability. Notice that the verification by refinement given above does not require identifying low-level actions in the implementation as linearization points, which is the difficult (and sometimes even impossible) task. In fact, the verification can be automatically carried out without any special knowledge about the implementation beyond knowing the implementation code.

## 5 SNZI Model

In order to prove that SNZI algorithm is a linearizable implementation, we model its specification and implementation in extended CSP, and then verify that the implementation refines the specification.

Fig. 4 shows the abstract specification model with  $P$  processes. Process *ArriveA* and *DepartA* consist of invocation event, linearization event  $\tau$  and response event. Process

```

ArriveA(i) = arrive_inv.i → τ{surplus++; }
           → arrive_res.i → Skip;
DepartA(i) = depart_inv.i → τ{surplus--; }
           → depart_res.i → Skip;
QueryA()  = query.(surplus > 0) → QueryA();
ProcessA(i) = ArriveA(i); DepartA(i); ProcessA(i)
SNZIA()    = (||| x : {0..P-1}@ProcessA(x))\{τ}
           ||| QueryA();

```

**Figure 4. Abstract specification model**

```

ArriveI(p, n) = arrive_inv.p →
              if (n == 0) ArriveR(p) else Arrive(p, n);
              arrive_res.p → Skip;
DepartI(p, n) = depart_inv.p →
              if (n == 0) DepartR(p) else Depart(p, n);
              depart_res.p → Skip;
Process(i)    = □ x : {0..N-1}@
              (ArriveI(i, x); DepartI(i, x));
Query()       = query.I → Query();
SNZI()        = (||| x : {0..P-1}@Process(x))\{τ}
              ||| Query();

```

**Figure 5. Concrete implementation model**

*QueryA* recursively reads whether *surplus* is greater than zero or not. *ProcessA* models the behavior of a process, i.e., repeatedly performs an *ArriveA* followed by a *DepartA*. *SNZIA*<sup>3</sup> interleaves all *ProcessAs* and *QueryA* and hides the  $\tau$  events (i.e., the linearization events).

The basic structure of the implementation (the details of *Arrive* and *Depart* operations are skipped) is showed in Fig. 5. To initialize the rooted tree in the implementation, a size  $N$  array named *node* is created to store SNZI objects. The root is *node*[0], and for  $0 < i < N$ , the parent of *node*[ $i$ ] is *node*[ $\frac{i-1}{2}$ ]. Since  $P$  processes may visit the same node concurrently, an  $N \times P$  array is introduced to store the local variables within an operation of  $P$  processes visiting  $N$  nodes. The full implementation model can be found in the built-in examples of PAT [12] (<http://pat.comp.nus.edu.sg>).

A process could visit any node at any time, i.e., which node a process chooses to visit is decided by external environment. Thus, external choice  $\square$  is used to represent a process visiting a node randomly. *ArriveI*( $p, n$ ) represents the process  $p$  arriving at the node  $n$ . If  $n = 0$  (the visiting node is the root), then it starts process *ArriveR* which captures how a process enters the root. Otherwise, it starts process *Arrive* which captures how a process arrives a hierarchical node. So does *DepartI*. Due to space constraints, we show the resulting code only for *Depart* operation at the

```

1. DepartR(p) =
2. τ{c[p] = C[0]; a[p] = A; v[p] = V[0]; } →
3. if (c[p] == C[0] && a[p] == A && v[p] == V[0]) {
4.   τ{C[0] = c[p] - 1; A = false; V[0] = v[p]; } →
5.   if (c[p] > 1) {τ → Skip}
6.   else {τ → DepartLoop(p)}
7. } else {τ → DepartR(p)};
8. DepartLoop(p) = τ{counts[p] = count; } →
9. if (v[p] != V[0]) {τ → Skip}
10. else {
11.   if (counts[p] != count) {τ → DepartLoop(p)}
12.   else {τ{I = false; count++; } → Skip}
13. };

```

**Figure 6. Depart operation on root node**

root in Fig. 6. The original algorithm of *Depart* includes two-fold loop statements. Each loop is modeled as a recursively defined process. *DepartR* process models the outer loop, while *DepartLoop* models the inner loop. The original  $X$  and  $x$  are both structured variables composed of three simple variables (represented respectively by  $(C, A, V)$  and  $(c, a, v)$ ). An atomic and invisible event  $\tau$  containing the assignment statements of  $c, a$  and  $v$  represents the assignment of  $x$  on line 2. Similar is  $X$  on line 4. For line 5, 6 and 7, another  $\tau$  is added between if/else condition and the first event of true/false branch to prevent them from executing in one atomic step. *DepartLoop* contains a pair of *LL/SC* primitives. The value of *counter* is recorded when performing *LL* (line 8). Then when the process attempts *SC*, it checks whether the recorded value is same as the current value of *counter* (line 11). If they are not equal, *DepartLoop* is repeatedly invoked (line 11). Otherwise, the process assigns *false* to  $I$  and then performs *Skip* event to return control to the invoking process (line 12).

## 6 Verification and Experimental Result

Based on Theorem 2, automatic refinement checking allows us to verify the linearizability of SNZI algorithm. PAT [10] supports different notions of refinements based on different semantics. A refinement checking algorithm (inspired by the one implemented in FDR [9] but extended with partial order reduction) is used to perform refinement checking on-the-fly. The key idea is to establish a (weak) simulation relationship from the specification to the implementation. We remark that FDR does not support shared variables/arrays, and therefore, is not easily applicable. Another candidate tool is the SPIN model checker, which supports verification of LTL properties. Nonetheless, formalization linearizability as LTL formulae results in large LTL formulae and thus not feasible for verification.

<sup>3</sup>|||  $x : \{1..N\}@P(x)$  is same as  $P(1) ||| \dots ||| P(N)$ , similarly for  $\square$ .

We have experimented SNZI on PAT for different number of processes and tree nodes. The table below summarizes the results, where ‘-’ means infeasible, and ‘POR’ means partial order reduction. The testbed is a PC with 2.83GHz Intel Q9550 CPU and 4 GB memory.

Setting		Result without POR		Result with POR	
#Proc	#Node	Time(sec)	#States	Time(sec)	#States
2	2	23.3	28163	17.1	23828
2	3	73.6	62753	41.4	52779
2	4	393	376342	157	173694
2	5	1298	712857	322	341845
2	6	-	-	496	485156
3	2	-	-	6214	8451568

The number of states and running time increase rapidly with data size, and especially the number of processes. This conform to theoretical results [1]: model checking linearizability is in EXPSpace. We have employed several optimization techniques to improve scalability. First, we use partial order reduction to effectively reduce the search space and running time. Second, we manually combined sequences of local actions into atomic blocks, such as organizing consecutive events which only cope with local variables into one single  $\tau$  event. Third, we specified every operation using a minimum number of processes, in order not to generate multiple equivalent states as different parameterized processes containing the same events. Overall, our approach is effective to handle big models like SNZI.

## 7 Related Work and Conclusion

The idea of refinement has been explored by Alur, et al. [1] to show that linearizability can be cast as containment of two regular languages. Our definition of linearizability on refinement is more general, regardless of the modeling language and knowledge of linearization points.

Formal verification of linearizability is a much studied research area. There are various approaches in the literature. Verification using theorem provers is another approach [4], where algorithms are proved to be linearizable by using simulation between input/output automata modeling the behavior of an abstract set and the implementation. However, theorem prover based approach is not automatic. Conversion to IO automata and use of PVS require strong expertise. Wang and Stoller [14] present a static analysis that verifies linearizability for an unbounded number of threads. Their approach detects certain coding patterns, hence is not complete (i.e., not applicable to SNZI algorithm). Amit et al. [2] presented a shape difference abstraction that tracks the difference between two heaps. The main limitation of this approach is that users need to provide linearization points, which is generally unknown. A buggy design may have no linearization points at all. In [13], Vechev and Yahav provided two methods for linearizability checking. One

method requires user annotations for linearization points. The other is fully automatic but inefficient (The worse case time is exponential in the length of the history). As a result, the number of operations they can check is only 2 or 3. In contrast, our approach handles all possible interleaving of operations given sizes of the shared objects.

In this work, we expressed linearizability using refinement relation. By using this definition, we have successfully verified the SNZI algorithms for the first time. We have shown that the refinement checking algorithm behind PAT allows us to successfully verify complicated concurrent algorithms without the knowledge of linearization points. During the analysis, we have faced the infamous state explosion problem. In future, we will explore how to combine different state space reduction techniques and parameterized refinement checking for infinite number of processes.

## References

- [1] R. Alur, K. Mcmillan, and D. Peled. Model-checking of correctness conditions for concurrent objects. In *LICS 96*, pages 219–228. IEEE, 1996.
- [2] D. Amit, N. Rinetzky, T. Reps, M. Sagiv, and E. Yahav. Comparison under abstraction for verifying linearizability. In *CAV 07*, pages 477–490. Springer, 2007.
- [3] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. John Wiley & Sons, Inc., Publication, 2nd edition, 2004.
- [4] S. Doherty, L. Groves, V. Luchangco, and M. Moir. Formal verification of a practical lock-free queue algorithm. In *FORTE 04*, pages 97–114. Springer, 2004.
- [5] F. Ellen, Y. Lev, V. Luchangco, and M. Moir. SNZI: Scalable NonZero Indicators. In *PODC 07*, pages 13–22, 2007.
- [6] M. Herlihy and J. M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.
- [7] C. A. R. Hoare. *Communicating Sequential Processes*. International Series in Computer Science. Prentice-Hall, 1985.
- [8] Y. Liu, W. Chen, Y. A. Liu, and J. Sun. Model Checking Linearizability via Refinement. Technical Report MSR-TR-2009-29, Microsoft Research Asia, March 2009. <http://research.microsoft.com/apps/pubs/?id=79938>.
- [9] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.
- [10] J. Sun, Y. Liu, and J. S. Dong. Model Checking CSP Revisited: Introducing a Process Analysis Toolkit. In *ISoLA 08*, pages 307–322. Springer, 2008.
- [11] J. Sun, Y. Liu, J. S. Dong, and C. Q. Chen. Integrating Specification and Programs for System Modeling and Verification. In *TASE 09*, 2009. (To appear).
- [12] J. Sun, Y. Liu, J. S. Dong, and J. Pang. PAT: Towards Flexible Verification under Fairness. In *CAV 09*, 2009. (To appear).
- [13] M. Vechev and E. Yahav. Deriving linearizable fine-grained concurrent objects. In *PLDI 08*, pages 125–135, 2008.
- [14] L. Wang and S. Stoller. Static analysis of atomicity for programs with non-blocking synchronization. In *PPoPP 05*, pages 61–71. ACM Press New York, NY, USA, 2005.

# Detecting Defects with an Interactive Code Review Tool Based on Visualisation and Machine Learning

Stefan Axelsson  
Blekinge Institute of Technology  
stefan.axelsson@bth.se

Dejan Baca  
dejan.baca@bth.se

Robert Feldt  
robert.feldt@bth.se

Darius Sidlauskas  
dsidlauskas@gmail.com

Denis Kacan  
denis.kacan@gmail.com

## Abstract

*Code review is often suggested as a means of improving code quality. Since humans are poor at repetitive tasks, some form of tool support is valuable. To that end we developed a prototype tool to illustrate the novel idea of applying machine learning (based on Normalised Compression Distance) to the problem of static analysis of source code. Since this tool learns by example, it is trivially programmer adaptable. As machine learning algorithms are notoriously difficult to understand operationally (they are opaque) we applied information visualisation to the results of the learner. In order to validate the approach we applied the prototype to source code from the open-source project Samba and from an industrial, telecom software system. Our results showed that the tool did indeed correctly find and classify problematic sections of code based on training examples.*

## 1. Introduction

An important part of ensuring code quality is code review [11]. Code review, which in effect is a form of manual static analysis of the code, is important especially when it comes to finding code that hides problems that are more difficult to find using some form of dynamic analysis, for example relating to non-functional requirements such as security. However, code review is problematic in that human operators are especially fallible when faced with a repetitive monotonous task, such as going through reams and reams of source code [12]. Some form of automatic static code analysis is then suggested. Static analysis have many inherent advantages; it can be applied early in the development process to provide early fault detection, the code does not have to be fully functional, or even runnable and test cases do not have to be developed.

To support manual code review we have developed a prototype tool—called The Code Distance Visualiser—to help the operator find problematic sections of code. The prototype is based on the novel idea of applying a machine learning technique, using Normalised Compression Distance (NCD) calculations, to provide the operator with an interactive, supervised self learning static analysis tool. This also makes the tool trivially programmer adaptable, that is, the tool will adapt to the task at hand as a consequence of the applied training. This is important in that traditional tools that are programmer adaptable (such as *Coverity* [8]) seldom are applied in that capacity [7].

Another advantage of machine learning is its capacity for generalising from a set of examples. Correctly applied machine learning has the capacity to *surprise* the operator by producing results that were previously unanticipated, while still being relevant and correct. However, machine learners are notoriously difficult to understand operationally, that is, they are opaque; it is difficult to understand when they are operating optimally and why they produce the results they do. To combat this problem we have applied information visualisation to the learner.

In the remainder, section 2 present related work, while sections 3 and 4 present Normalized Compression Distance and how we parse and represent the source code in order to apply it. In section 5 our tool and its visualisation capability are presented briefly. Section 6 present the experiments we have conducted with the tool and the results we have obtained. In section 7 we discuss the results. Finally, section 8 points to future work and concludes.

## 2. Related work

The previous work that is closest overall to the work presented here is probably that of Brun and Ernst [3]. They implemented a tool that is trained using machine learning techniques to identify program properties which indicate

errors. There are two primary differences between our approach and their work. First, they use dynamic analysis to extract semantic properties of the program's computation, whereas we use static analysis. Second, their tool uses a classical batch-learning approach, in which a fixed quantity of manually labelled training data is collected at the start of the learning process. In contrast, the focus of our work is on incremental learning by (potentially a series of) manual user interactions.

Statistical machine learning techniques (Markov modelling, bootstrapping) were successfully applied on classifying the program's behaviour by Bowring, et. al. [2]. There the classifier was trained incrementally to map execution statistics such as branch profiles to a label of program behaviour such as *pass* or *fail*.

So far we have not found any static code analyser proper similar to ours. The one most related is probably the one by Kremenek, et. al. [14], where a feedback rank scheme is used. Correlation among reports (errors reported by analyser) is represented in a probabilistic model Bayesian network. Then network is trained during interactive inspection of reports and probabilities for uninspected reports are recalculated. This approach primarily attacks the false positives problem in static code analysis and learning techniques are applied just to perform error ranking. In comparison, we use machine learning explicitly for finding errors and apply it directly to static analysis.

The successful applications of normalised information distance to various problem domains are too numerous to detail here, but it has been previously proposed for software quality-related tasks [9]. However, probably the closest application of normalised information distance to the one proposed here is in plagiarism detection within student programming assignments. The *Software Integrity Diagnosis (SID)* system [5] uses a variant of the normalized information distance (NID) to measure the similarity between two source code files. The plagiarism detector parses the source code much as our approach does. However, the SID system does not provide any interactivity for the user. The process also acts as a black box. This is not surprising, as the aims of their technique and ours are completely different. Indeed, keeping the analysis opaque might be done on purpose to avoid exposing inner system's state to the cheaters.

### 3. Normalised Compression Distance (NCD)

The problem is one of supervised machine learning, i.e. the operator will select sections of code to train the machine learner with. Of the several available algorithms we have chosen to base our machine learner on a fairly recent algorithm that computes distances between arbitrary data vectors: *Normalised Compression Distance (NCD)* [6] as it is generally applicable [10], parameter free [13], noise re-

sistant [4] and demonstrated theoretically optimal [17]. The NCD is an approximation to the uncomputable *Normalised Information Distance*, that is based on the notion of Kolmogorov Complexity.

NCD is based on the idea that by using a compression algorithm on data vectors (in whatever shape or form these may be) both individually and together, we will receive a measure of how distant they are. The better the combination of the two vectors compress, compared to how the individual vectors compress on their own (normalised to remove differences in length between the set of all vectors), the more similar they are. More formally, NCD is a metric:

$$NCD(x, y) = \frac{C(x, y) - \min(C(x), C(y))}{\max(C(x), C(y))}$$

where  $C(x)$  is the compressed length of  $x$  and  $C(x, y)$  the compressed length of  $x$  concatenated with  $y$ .

In order to apply this metric as a supervised learner one selects features of the input data to train on and then calculates the distances from instances in the two (or more) sets of the selected features. Our tool contains two possible training sets; one *bad* containing undesirable features, and one *good* containing desirable features. The tool then calculates distance from all bad and all good features and presents the results to the operator. The classification is by the closest example in either set, e.g. a code feature that is closest in distance to one particular instance in the *bad* set is classified as *bad*, and vice versa, i.e. the machine learning algorithm proper is *k-nearest-neighbour* with  $k = 1$ .

### 4. Parsing

The machine learning algorithm could not be successfully applied to the source code as is, as usual, feature vectors had to be selected [15]. We run the code under study through a parser to produce our features. In the prototype we have chosen the *C*-language, as parsers (and code with which to validate our approach) are readily available. However, besides the particulars of parsing and feature selection we see no reason that our approach would not be applicable to other languages, even those dissimilar to *C*. For the prototype we have chosen the freely available parser *Sparse*.<sup>1</sup>

The task of how to do feature selection of source code is not at all well studied. Based on our—admittedly limited—experience we have chosen to first parse the code and then emit an adapted textual representation of the source code that is fed back to the machine learning step. Based on our experiences we've chosen the following strategies for parsing and presenting the source code.

In order to make the source code amenable to machine learning, all whitespace and a few other “uninteresting” reserved words (such as semi colons, angle

<sup>1</sup><http://www.kernel.org/pub/software/devel/sparse/>

brackets etc.) are removed. A stickier problem with a textual representation, stemming from dealing with a machine learner that can't differentiate between different data types, is that longer strings will fool the learner into thinking they carry more information compared to shorter strings. That means that for example *the\_very\_long\_identifier\_that\_never\_ends*, e.g. used as a variable in the source code, will carry relatively more weight than a shorter identifier, say *foo*. The same is true of reserved words. In order to alleviate this problem the parser can be set to exchange all variable names with a two character abbreviation (the same two character substitution is maintained for the same variable as far as is possible). A third problem, opposite of the previous, is that certain C operators are too *short* and too similar to each other to be distinct enough in a pure textual representation. Examples are `==` and `=`, which are known from experience to be difficult to tell apart. In order to remedy this problem these operators are exchanged with unique textual representations that are longer and more different from each other to give the machine learner more to latch on to.

Since our chosen algorithm does not naturally handle sequences of varying length we have to address the unit-of-analysis problem. One approach is to slide a window of a certain length over the features under study, but this tends towards a combinatorial explosion that we can ill afford, and it also performed poorly in preliminary tests with short but reasonable window sizes. Instead we have chosen to implement a varying level of detail that the user can choose; the statement (basically a line of source code) and basic block level (code between curly braces). The basic block can contain a *while/if* etc. statement introducing the block.

## 5. The Code Distance Visualizer

We will now describe how the preceding pieces were put together to form a tool that enables a user to select sections of source code to train the detector on and view the results of the training on source code (classifying). As described in the previous section, the naive NCD classification classifies code fragments into faulty or correct. However, when a training process is being presented as a *black box*, it is difficult to ascertain how the classifier is being trained. In order to be able to judge the quality of the output, the training process has to be transparent. Once the user has visual access to the internal state of the classifier, she can more precisely understand what the learner is actually learning and then interactively guide it by marking additional code fragments on screen as faulty or correct. Afterwards, all code fragments must be classified and visually marked. Unfortunately a lack of space precludes us from going into more detail on how this is done more exactly, save to say that the selected code fragments are heat-

mapped [16] whereby the code fragment is coloured from red via yellow to green. The spectrum of colour depends on the NCD value of a particular code fragment and whether it is more likely to be faulty (red colour) or correct (green colour). This approach is inspired by that of Axelsson [1].

In order to present the ideas implemented in the prototype we present the user interface in figure 1. It can be divided into a few major parts: the original source view, the adapted source view, the list of training instances, the ranking view and the code fragment information view.

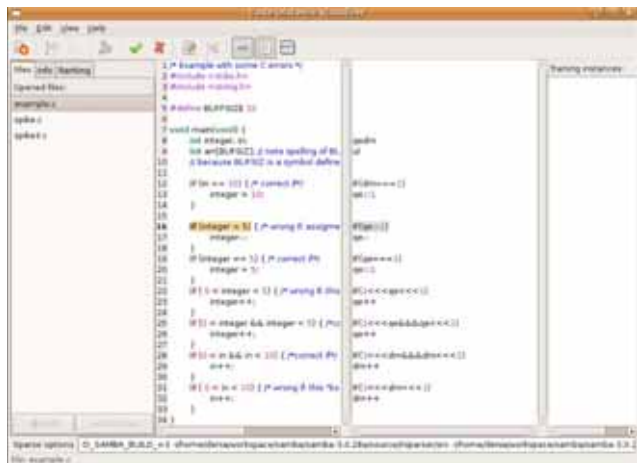


Figure 1. The main window of the prototype, no training.

A code fragment can be composed of structures such as a code block, individual statements, expressions, and mixture of the above. The user has a wide range of choices available for selecting code fragments of various lengths. For instance, in figure 1 one can see how the corresponding individual statements are selected in both original (`if (integer = 5)`) and adapted views (`IF(qe:::1)`).

After having trained the analyser on a particular code fragment the user may call up a list of the top ranked faulty or correct code fragments as classified by the machine learner. The user has a choice of which types of code fragments (individual statements, code blocks and expression blocks) that will be ranked.

The status of each code fragment depends on the distance to the closest training instance. Thus, in order to understand why a particular code fragment was classified as being faulty or correct, the user has to see how it is related to all the training instances.

The tool is available under the GPL on [sourceforge](http://sourceforge).

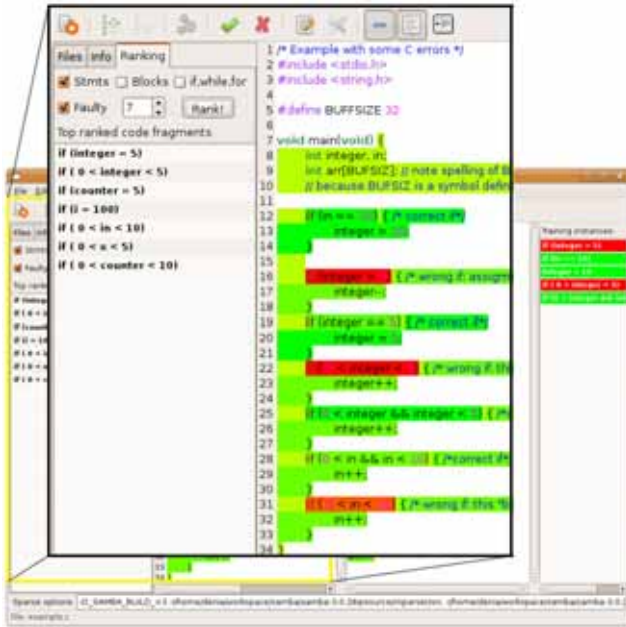


Figure 2. The ranking view with the top seven faulty individual statements.

## 6. Experimental Validation

A serious problem for researchers wishing to experiment with software validation techniques is the problem of locating suitable experimental samples. We even need multiple versions of the same software (erroneous and error-free). Obtaining such samples is nontrivial. To empirically evaluate the proposed technique, we applied our prototype to experimental samples from two real-world sources. Our goal for the experiments with our prototype was to obtain information about the effectiveness of the proposed technique for fault detection. The question of how effective the visualisation approach was in conveying information to the users that she would otherwise not have, will be left for further study.

We analysed a number of open source projects and chose the open source project Samba, at the time stable release Samba 3.0.28a, as that had the most accessible defect database. In addition a closed source commercial telecommunications software was selected. Both are server software that are written in the C language and each is in the million lines of source code range. For each project a subset of the source code was selected and analysed. Approximately 20000 lines of code was used during the analysis, 13000 from Samba and 7000 from the commercial product. The commercial product handles large quantities of network data and performs extensive computation on that data. Due to corporate policy the product must unfortunately remain unnamed, as must the company. The decision of which

freely available project to include was not rigorous in the sense that we made our choice depending on first impressions, thus other software may also be suitable. The telecom software was chosen because we already had access to and are very familiar with it. By using software from two different domains we aim to improve the general applicability of the results of the experiments.

The experiment was set up as a proof of principle; we applied the Code Distance Visualiser on code fragments that were analysed in advance, fragments where we knew what the answer already should be. We thus trawled the Samba bug database for defect data. When good experimental data was available, the prototype was trained by feeding it some known faulty code fragments and their corrected versions. We then checked whether the prototype correctly identified the remaining faults. Since an interactive tool with feedback was tested, several possible strategies for (re)training presented themselves. From our experience, a good strategy was to start training the prototype with one faulty code fragment and its corrected version, as it then learned the precise differences between faulty and correct code. We call this two-instance training *initial training* as it was the first training performed in the experiments. Also the ranking feature was used to improve the analysis process. For example, in Samba's code (tagged as SAMBA-3-0-RELEASE) a memory leak bug was removed in nine places (revision 21755 of file net\_rpc.c). Thus, we retrieved this and previous revisions, trained the prototype with one of the faulty code blocks and its corrected version (with the memory leak removed). Then we used the ranking feature to see the top nine (as we knew there were nine) faulty code blocks and checked whether all the remaining memory leaks were present among them. If some of them were missing, because of some false positives being present in the top ranking, we continued the training. Additional training involved marking the false positive at the highest position in the ranked list as correct. Usually this made the top ranked and similar false positives disappear from the ranking, freeing up their positions for other code fragments. From the two products that were examined, four distinctly different types of code defects were identified and used during the investigation. They were:

**String overflows** Defects where during various string operations a target char array overflowed. These defects very often lead to a security vulnerability. The most common string overflow is created by using insecure library functions such as `strcpy()`. An often used solution is to perform input validation or using a more secure function (such as `asstrncpy()`).

**Null pointer references** These are defects where a pointer that might be NULL were used without prior checking. These defects result in segmentation faults or other un-



predictable behaviours. The main cause is lack of error checking or using already freed memory pointers.

**Memory leaks** These result from dynamically allocated memory that is not freed. Dynamically allocated memory needs to be de-allocated before its pointers goes out of scope. These defects are often introduced by carelessness but can also arise from special cases when the program returns unexpectedly from a function.

**Incorrect API usage** A fault that stems from libraries and functions that are used correctly on the face of it but do not perform the task the programmer intended. These coding defects often occur many times in the code and a simple textual search could detect them, but instances of correct usage would drown the true positives in the search results.

Since we chose to investigate the same types of flaws in both products we will present the aggregated results. Where the discussion refers to only one code base that will be noted. Table 1 summarise the data from the investigation. The first column states what fault type is examined. The *total* column shows how many known faults of that type exist in the data. The *CI* are correctly identified faults while *FN* are false negatives; faults that were not detected by the tool. Correctly identified and false negative faults are shown in two separate columns, after initial training and after additional training as described in the previous section. The *improvement* column lists the relative improvement in detection rate the tool displayed after additional training compared to the initial training.

Name	Total	Init Train		Addtl		Imprvmt
		CI	FN	CI	FN	
Str ovfl	8	4	4	8	0	100
Null p ref	44	22	22	29	15	24
Mem-leak	12	7	5	9	3	22
Inc API	14	9	5	14	0	55

**Table 1. Experimental results**

The *string overflows* are string copy operations that in some cases also create a security vulnerability. The known string overflow bugs had been reported by both testers and the static code analyser, *Coverity Prevent*.<sup>2</sup>

All the Null pointer reference bugs were originally reported from a static code analyser, either *KlocWork*<sup>3</sup> or *Coverity*. While they had a bug report none was from a tester that had experienced a crash (this relates to both tested products). These defects often propagate and turn up later during execution as a segmentations fault and are difficult for the tester to ascribe a particular section of code.

<sup>2</sup><http://www.coverity.com>

<sup>3</sup><http://www.klocwork.com/>

There were two types of memory leaks of different complexity in the projects. The simpler memory leaks were instances where the programmer forgot to free or in this case call `shutdown` to free allocated memory.

The incorrect API usage involved four commits with fixes. It was a correction to the caller `unistr2_to_ascii`, as the *maxlen* parameter should be set to the size of the destination, not to the size of the source string. The entire procedure took less than 10 minutes.

## 7. Discussion

Our results show that the tool can be used to effectively detect security defects in real-world source code. Even the initial training on a single faulty and correct example lead to the correct identification of 53.8% of the 78 identified defects. More extensive training, utilising the ranking feature of our tool, lead to the correct identification of 76.9% of the total number of defects. Detected defects are not simple in the way that a traditional find search in the source code would easily identify the patterns in the defects. For example, in the incorrect API case, 216 different instances of the API call was present but only in 14 of them is the API used incorrectly. With our tool all 14 of these defects were detected in only 10 minutes and there were no false negatives.

We focused on following a consistent training strategy for our results to be comparable across faults and projects studied. We found that keeping a balance between the examples of faulty and correct instances worked best. This relates to the machine learning concept of overfitting where the learner might pick up on some very specific feature and become over trained in one category.

The choice of *unit-of-analysis* (how large a code fragment we train on) affects the size and scope of the defects we can detect. Defects that depend on an interplay of dependent parts of the code might be hard to detect unless its individual parts are unique enough that they can be detected in isolation. But at the same time, isolated defects without the surrounding code do not present enough unique data to be detected, as was shown by the NULL pointer references defects. We could possibly find alternative ways to extract features that could extend the reach of our proposed method.

While we have not formally verified the usability of the prototype, it has had two users who were not part of the development of the tool. They both spontaneously reported that the visualisation and interactivity were both worthwhile additions and made working with the automatic classifier more pleasant and less opaque, even though there is still room for improvement in that respect.

## 8. Conclusion and Future work

We have only really scratched the surface of the possibilities of this technique. Obvious future work includes: study of how to extract features useful for machine learning from a static representation of source code, as the present approach is rather naive and no real extraction of semantic features takes place, investigating the applicability of different machine learning algorithms to static analysis, investigating the effect the visualisation has on the usability of the tool, to name just a few.

Moving further from the area of static analysis; by modifying the approach we have presented here, the approach could possibly be applied at even earlier stages of software development projects. Potentially it could be used on design, requirements or specification documents to highlight decisions that might lead to security or other software quality challenges.

In conclusion: We have developed a prototype tool that applies the novel idea of machine learning to aid in code review. The tool uses information visualisation techniques to try to alleviate the problem of machine learning techniques being opaque to the user, i.e. it is difficult to ascertain why they perform a particular classification in the manner they do. While much work still remains to be done in this domain, initial results were promising. When we applied the prototype to two different code bases, the tool successfully managed to identify problematic sections based on the examples given, without producing too many false alarms. Furthermore, during the experiment the tool managed to generalise from the example of the faulty `strcpy` operation so that it detected that a similar `strcpy` fault was also erroneous. The ability to generalise from given examples are a prime reason to apply machine learning to a problem. The tool also managed to identify a situation where features were *missing* from the code, which is an important class of faults.

## References

- [1] S. Axelsson. Combining a bayesian classifier with visualisation: Understanding the IDS. In C. Brodley, P. Chan, R. Lippman, and B. Yurcik, editors, *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security (VizSec'04)*, pages 99–108, Washington DC, USA, 29 Oct. 2004. ACM Press. Held in conjunction with the Eleventh ACM Conference on Computer and Communications Security.
- [2] J. F. Bowring, J. M. Rehg, and M. J. Harrold. Active learning for automatic classification of software behavior. *SIGSOFT Softw. Eng. Notes*, 29(4):195–205, 2004.
- [3] Y. Brun and M. D. Ernst. Finding latent code errors via machine learning over program executions. In *ICSE'04, Proceedings of the 26th International Conference on Software Engineering*, pages 480–490, Edinburgh, Scotland, May 26–28, 2004.
- [4] M. Cebrian, M. Alfonseca, and A. Ortega. The normalized compression distance is resistant to noise. *Information Theory, IEEE Transactions on*, 53(5):1895–1900, May 2007.
- [5] X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker. Shared information and program plagiarism detection. *Information Theory, IEEE Transactions on*, 50(7):1545–1551, July 2004.
- [6] R. Cilibrasi. *Statistical Inference Through Data Compression*. PhD thesis, Institute for Logic, Language and Computation Universiteit van Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, 2007. <http://www.illc.uva.nl/>.
- [7] D. Engler. Weird things that surprise academics trying to commercialize a static checking tool. Invited talk at SPIN'05 and CONCUR'05, 2004. <http://www.stanford.edu/engler/spin05-coverage.pdf>.
- [8] D. Engler, B. Chelf, A. Chou, and S. Hallem. Checking system rules using system-specific, programmer-written compiler extensions. In *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI 2000)*, San Diego, California, USA, Oct. 2000. USENIX.
- [9] R. Feldt, R. Torkar, T. Gorschek, and W. Afzal. Searching for cognitively diverse tests: Towards universal test diversity metrics. In *Proceedings of the First Workshop on Search-Based Software Testing*, pages 178–186, Lillehammer, Norway, Apr. 2008.
- [10] P. Ferragina, R. Giancarlo, V. Greco, G. Manzini, and G. Valiente. Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. *BMC Bioinformatics*, 8(1):252, 2007.
- [11] M. Höst and C. Johansson. Evaluation of code review methods through interviews and experimentation. *Journal of Systems and Software*, 52(2):113–120, Apr. 2000.
- [12] M. Howard. A process for performing security code reviews. *IEEE Security & Privacy*, 4(4):74–79, July 2006.
- [13] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, New York, NY, USA, 2004. ACM.
- [14] T. Kremenek, K. Ashcraft, J. Yang, and D. Engler. Correlation exploitation in error ranking. In *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*, pages 83–93, New York, NY, USA, 2004. ACM.
- [15] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. ISBN 0-07-115467-1.
- [16] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, second edition, May 2001. ISBN 0-96-139214-2.
- [17] P. Vitanyi, F. Balbach, R. Cilibrasi, and M. Li. *Information Theory and Statistical Learning*, chapter Chapter 3. Springer-Verlag, 2008.

# Dynamic Test Profiles in Adaptive Random Testing: A Case Study

Huai Liu\*, Fei-Ching Kuo, and Tsong Yueh Chen

Centre for Software Analysis and Testing, Swinburne University of Technology, Australia

{hliu, dkuo, tychen}@swin.edu.au

## Abstract

*Random testing (RT) is a basic software testing method. When used to detect software failures, RT usually generates random test cases according to a uniform distribution. Adaptive random testing (ART) is an innovative approach to enhancing the failure-detection capability of RT. Most ART algorithms are composed of two independent processes, namely the candidate generation process and the test case identification process. In these ART algorithms, some program inputs are first randomly generated as the test case candidates; then test cases are identified from these candidates in order to ensure an even spread of test cases across the input domain. Most previous studies on ART focused on the enhancement of the test case identification process, while using the uniform distribution in the candidate generation process. A recent study has shown that using a dynamic test profile in the candidate generation process can also improve the failure-detection capability of ART. In this paper, we develop various test profiles and integrate them with the test case identification process of a particular ART algorithm, namely fixed-size-candidate-set ART. It is observed that all these test profiles can significantly improve the failure-detection capability of ART.*

## 1. Introduction

*Random testing (RT)*, a fundamental software testing approach [13], can be used as both a reliability assessment technique [15] and a debug testing method (that is, a method aiming at detecting software failures so that program bugs can be removed [11]). In the context of debug testing, RT usually generates *test cases* (that is, program inputs for testing) based on a uniform distribution from the whole *input domain* (that is, the set of all possible inputs). In other words, all program inputs have the same probability to be generated as test cases.

Although RT has been used in various areas to detect software failures [14, 16], some researchers considered RT as ineffective because RT simply detects failures by chance [13]. Many independent studies [1, 10] have shown that failure-causing inputs (that is, inputs that cause the program under test to exhibit failure behaviors) tend to cluster into contiguous regions (known as *failure regions* [1]) in the input domain. Chen *et al.* [8] made use of such a common characteristic of failure-causing inputs to improve the failure-detection capability of RT. They proposed a novel approach, namely *adaptive random testing (ART)*, where test cases are not only randomly generated, but also evenly spread over the input domain. The basic intuition of ART, that is, the even spread of random test cases, is essentially a form of test cases' diversity across the input domain [5]. In fact, the diversity of test cases is the key concept for most test case selection strategies (such as coverage-based testing methods [17]). ART has been used for testing various programs, from software with numeric inputs [2, 8] to that with complex non-numeric inputs [9].

Various ART algorithms have been proposed to achieve the goal of evenly spreading test cases, such as *fixed-sized-candidate-set ART (FSCS-ART)* [8], *restricted random testing (RRT)* [2] and *lattice-based ART* [12]. Most ART algorithms consist of two independent processes – (a) *candidate generation process*, where some program inputs are randomly generated as *test case candidates*, or briefly *candidates*, and (b) *test case identification process*, where some test case identification criteria are applied to identify test cases amongst these candidates such that the identified test cases are evenly spread over the input domain. Different test case identification criteria lead to different ART algorithms. Previous studies have shown that ART can detect failures more effectively than RT in many cases.

Most studies on ART used the uniform distribution as the test profile in the candidate generation process. Recently, Chen *et al.* [4] proposed a new approach, namely *ART with dynamic non-uniform candidate dis-*

---

\*Corresponding author

*tribution* (ART-DNC). In ART-DNC, the candidate generation process is no longer conducted based on a uniform distribution, but on a dynamic non-uniform test profile. They selected one particular test profile and integrated such a profile with the test case identification processes of FSCS-ART and RRT algorithms. Their simulation studies showed that using the new test profile can significantly improve the effectiveness of the original ART algorithms.

In this paper, we further investigate into various test profiles that may be suitable for ART, and combine them with the test case identification process of FSCS-ART. We attempt to see whether and to what extent these profiles can enhance the failure-detection capability of ART. The rest of the paper is organized as follows. Section 2 introduces the background information on FSCS-ART and ART-DNC. In Section 3, we propose different test profiles, and investigate the effectiveness of ART that uses these test profiles in the candidate generation process. Section 4 concludes the paper.

## 2. Background

Fixed-size-candidate-set ART (FSCS-ART) [8] is a typical ART algorithm. In FSCS-ART, two sets of test cases are maintained, namely the *executed set*  $E$  and the *candidate set*  $C$ .  $E$  is composed of all the previously executed test cases, while  $C$  contains a fixed number of test case candidates that are normally generated in a random manner according to a uniform distribution. A candidate in  $C$  is identified as the next test case if its nearest neighbor distance to  $E$  is the longest. The details of FSCS-ART algorithm can be found in [8]. In our study, the size of the candidate set is set to 10, as recommended in [8].

ART aims to evenly spread random test cases over the whole input domain, but no ART algorithm is guaranteed to achieve such a goal under all possible scenarios [3]. Most previous studies on ART focused on the enhancement of test case identification process, but kept using the uniform distribution in the candidate generation process. Chen *et al.* [4] recently proposed ART with dynamic non-uniform candidate distribution (ART-DNC), which uses a test profile different from the uniform distribution in the candidate generation process. The aim of the new test profile in ART-DNC is to improve the evenness of test case distribution, and thus enhance the failure-detection capability.

FSCS-ART algorithm normally has a bias of identifying test cases from the edge part of the input domain rather than from the centre, and such an *edge bias* results in a certain degree of uneven test case distribution. FSCS-ART-DNC [4] was developed to integrate a new test profile with the test case identification

process of FSCS-ART algorithm. It has been suggested that the test profile used in FSCS-ART-DNC should (i) be dynamic along the testing process, (ii) assign a higher probability to the candidates from the central part of the input domain than those from the edge part (namely, the *centre bias*); and (iii) have a symmetric probability distribution with respect to the centre of the input domain. Readers who are interested may refer to [4] for the details of how to implement FSCS-ART-DNC algorithm.

The failure-detection capability of ART is normally measured by F-measure, which refers to the expected number of test cases required to detect the first software failure. Most previous studies of ART [2, 6, 8] estimated the F-measure of ART (denoted by  $F_{ART}$  in the rest of the paper) via simulations. In order to simulate faulty programs, these simulations first predefine two basic features of a faulty program, namely *failure rate* (denoted by  $\theta$ , which refers to the ratio between the number of failure-causing inputs and the number of all possible inputs) and *failure pattern* (which refers to the failure regions together with their distribution over the input domain). The size and shape of the failure region can then be decided based on  $\theta$  and the failure pattern, and the location of the failure region is randomly chosen inside the input domain. After setting up these parameters, ART is applied until the first failure is detected (that is, a point is picked from the failure region), and the number of test cases that ART has generated will be recorded. Such a process is repeated until we can get a statistically reliable value of  $F_{ART}$ . The details of how to conduct simulations can be found in [6]. The improvement of ART over RT is always evaluated by the *ART F-ratio*  $= F_{ART}/F_{RT}$ , where  $F_{RT}$  denotes the F-measure of RT that is theoretically equal to  $1/\theta$ .

## 3. Effectiveness of FSCS-ART-DNC with various dynamic test profiles

There exist many distributions that have the features mentioned in Section 2 (that is, features (i) to (iii)). Chen *et al.* only selected one dynamic distribution profile to illustrate the new ART-DNC approach. In this study, we propose three other test profiles, namely *triangle*, *cosine* and *semicircle* profiles, for FSCS-ART-DNC. These profiles are named after the basic shapes of the curves of their probability density functions (pdf), which are given in Formulas (1), (2), and (3). Obviously, the probability distributions of these profiles can be adjusted by changing the value of the parameter  $\alpha$ .

**Triangle profile:**

$$f_X(x) = \begin{cases} 4\alpha x + (1 - \alpha) & , 0 \leq x < 0.5 \\ -4\alpha x + (1 + 3\alpha) & , 0.5 \leq x < 1 \\ 0 & , x < 0 \text{ or } x \geq 1 \end{cases} \quad (1)$$

where  $0 \leq \alpha \leq 1$ .

**Cosine profile:**

$$f_X(x) = \begin{cases} \alpha \sin \pi x + \left(1 - \frac{2\alpha}{\pi}\right), & 0 \leq x < 1 \\ 0, & x < 0 \text{ or } x \geq 1 \end{cases} \quad (2)$$

where  $0 \leq \alpha \leq \frac{\pi}{2}$ .

**Semicircle profile:**

$$f_X(x) = \begin{cases} \alpha \sqrt{1 - (2x - 1)^2} + \left(1 - \frac{\alpha\pi}{4}\right), & 0 \leq x < 1 \\ 0, & x < 0 \text{ or } x \geq 1 \end{cases} \quad (3)$$

where  $0 \leq \alpha \leq \frac{4}{\pi}$ .

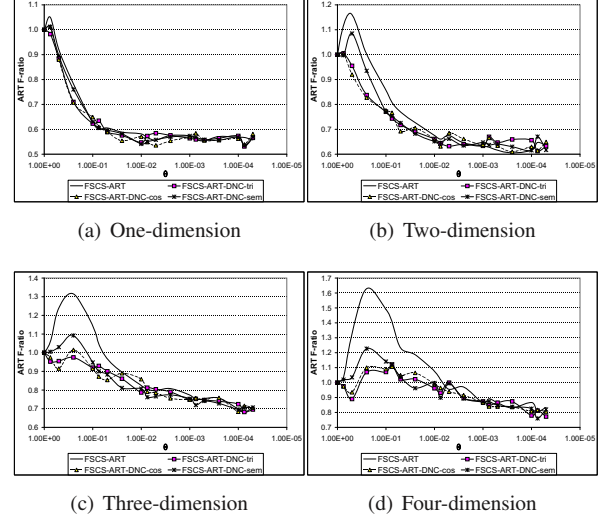
We attempted to integrate the above-mentioned three test profiles with the test case identification process of FSCS-ART, and then get three new ART algorithms, namely FSCS-ART-DNC with triangle, cosine and semicircle profiles. In these algorithms,  $\alpha$  in Formulas (1), (2), and (3) is dynamically adjusted along the testing process. In this paper, we use triangle profile to illustrate how to adjust the value of  $\alpha$ . For ease of illustration, assume that each dimension of input domain has the value range  $[0, 1)$ , and is equally divided into two subranges, namely the *centre subrange* consisting of  $[0.25, 0.75)$ ; and the *edge subrange* consisting of  $[0, 0.25)$  and  $[0.75, 1)$ . For each dimension, successively after each new test case is identified, the following three steps are conducted to adjust  $\alpha$ . First, we measure the ratio ( $r$ ) of the number of executed test cases from the edge subrange over the total number of executed test cases. Second, we calculate the probability ( $p$ ) of an element being generated from the centre subrange. From Formula (1), we can get

$$p = 0.25\alpha + 0.5 \quad (4)$$

Since  $0 \leq \alpha \leq 1$ ,  $p$  is within the value range  $[0.5, 0.75]$ . If  $0.5 \leq r \leq 0.75$ , we set  $p = r$ ; otherwise, we set  $p = 0.75$  (if  $r > 0.75$ ) or  $0.5$  (if  $r < 0.5$ ). Finally,  $\alpha$  can be determined from Formula 4.

We conducted a series of simulations to evaluate the failure-detection capabilities of these new FSCS-ART-DNC algorithms. In these simulations, the dimension of the input domain is one, two, three or four; the shape of the input domain is set as hyper-cube; the failure pattern is one hyper-cube randomly placed inside the input domain; and  $\theta$  is set from 1 to 0.00005. The simulation results are given in Figure 1, which also includes the previous results of the original FSCS-ART algorithm with uniform candidate distribution (denoted by ‘‘FSCS-ART’’ in the figure). The results of

three new FSCS-ART-DNC algorithms are represented by ‘‘FSCS-ART-DNC-tri’’, ‘‘FSCS-ART-DNC-cos’’ and ‘‘FSCS-ART-DNC-sem’’. In the figure, the x- and y-axes denote  $\theta$  and the ART F-ratio, respectively.



**Figure 1. Failure-detection capabilities of FSCS-ART-DNC with various test profiles**

Based on the simulation results, we can observe that all three FSCS-ART-DNC algorithms outperform the original FSCS-ART algorithm when the dimension of the input domain is high or  $\theta$  is high, and the performance improvement increases with the increase in dimension or  $\theta$ . For the cases of low dimension and low  $\theta$ , the failure-detection capability of the original FSCS-ART algorithm is very close to the theoretical bound that can be reached by an optimal testing method without prior information about the failure region’s location [7]. Therefore, it is expected that these FSCS-ART-DNC algorithms cannot significantly improve the performance of ART when dimension or  $\theta$  is low. Briefly speaking, using some proper dynamic test profiles in the candidate generation process does help to improve the failure-detection capability of ART, especially for the cases of high dimension and high  $\theta$ .

It can also be observed that there are some differences in the effectiveness of the three FSCS-ART-DNC algorithms. FSCS-ART-DNC-tri and FSCS-ART-DNC-cos always have similar failure-detection capabilities, but FSCS-ART-DNC-sem does not perform as well as the other two under the conditions of high dimension and high  $\theta$ . For example, when  $\theta = 0.25$  and the dimension is 4, the ART F-ratios of FSCS-ART-DNC-tri, FSCS-ART-DNC-cos, and FSCS-ART-DNC-sem are 1.07, 1.10, and 1.23, respectively. Such a phenomenon can be explained as follows. As shown in [3],

the original FSCS-ART algorithm has an edge bias, which becomes higher with the increase in dimension or  $\theta$ . The test profiles used in FSCS-ART-DNC all have a centre bias. From Formulas (1), (2), and (3), we can calculate that the triangle profile has the highest centre bias, followed by the cosine and semicircle profiles in descending order. When the dimension or  $\theta$  is low, the test case identification process of the original FSCS-ART algorithm does not deliver a very high degree of edge bias. In such a situation, all three test profiles can provide a sufficient degree of centre bias in the candidate generation process to offset the edge bias in the test case identification process. On the other hand, when the dimension and  $\theta$  are high, the low centre bias offered by the semicircle profile may not fully offset the extraordinary edge bias caused by the test case identification process. Therefore, it is intuitively expected that FSCS-ART-DNC-sem does not perform very well under the conditions of high dimension and high  $\theta$ . The similar performances of FSCS-ART-DNC-cos and FSCS-ART-DNC-tri imply that although the centre bias of the cosine profile is lower than that of the triangle profile, the former is sufficient to offset the edge bias in the test case identification process.

#### 4. Conclusions

Adaptive random testing (ART) was proposed to enhance the failure-detection capability of random testing as a debug testing method. Most previous studies have used the uniform distribution as the test profile for ART. A recent study has shown that using a dynamic test profile can further improve the failure-detection capability of ART. In this paper, we conducted some case studies on the application of three dynamic test profiles into ART algorithms. Simulation studies showed that all these three test profiles help to improve the failure-detection capability of ART.

Our experimental results also showed that different test profiles may bring out different failure-detection capabilities of ART. In the future work, we will analyze the statistical features of a variety of dynamic profiles and their impacts on the effectiveness of ART with various test case identification criteria. These investigations will provide new guidelines for how to develop and apply appropriate test profiles for different ART algorithms.

#### Acknowledgment

This research project is partially supported by an Australian Research Council Grant (DP0880295).

#### References

- [1] P. E. Ammann and J. C. Knight. Data diversity: an approach to software fault tolerance. *IEEE Transactions on Computers*, 37(4):418–425, 1988.

- [2] K. P. Chan, T. Y. Chen, and D. Towey. Restricted random testing: adaptive random testing by exclusion. *International Journal of Software Engineering and Knowledge Engineering*, 16(4):553–584, 2006.
- [3] T. Y. Chen, F.-C. Kuo, and H. Liu. On test case distributions of adaptive random testing. In *Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE 2007)*, pages 141–144, 2007.
- [4] T. Y. Chen, F.-C. Kuo, and H. Liu. Application of a failure driven test profile in random testing. *IEEE Transactions on Reliability*, 58(1):179–192, 2009.
- [5] T. Y. Chen, F.-C. Kuo, R. Merkel, and T. H. Tse. Adaptive random testing: the ART of test case diversity. Accepted to appear in *Journal of Systems and Software*.
- [6] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou. On favorable conditions for adaptive random testing. *International Journal of Software Engineering and Knowledge Engineering*, 17(6):805–825, 2007.
- [7] T. Y. Chen and R. Merkel. An upper bound on software testing effectiveness. *ACM Transactions on Software Engineering and Methodology*, 17(3):16:1–16:27, 2008.
- [8] T. Y. Chen, T. H. Tse, and Y. T. Yu. Proportional sampling strategy: a compendium and some insights. *Journal of Systems and Software*, 58(1):65–81, 2001.
- [9] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer. AR-TOO: adaptive random testing for object-oriented software. In *Proceedings of the 30th International Conference on Software Engineering (ICSE'08)*, pages 71–80. ACM Press, 2008.
- [10] G. B. Finelli. NASA software failure characterization experiments. *Reliability Engineering and System Safety*, 32(1–2):155–169, 1991.
- [11] P. G. Frankl, R. G. Hamlet, B. Littlewood, and L. Strigini. Evaluating testing methods by delivered reliability. *IEEE Transactions on Software Engineering*, 24(8):586–601, 1998.
- [12] J. Mayer. Lattice-based adaptive random testing. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005)*, pages 333–336, New York, USA, 2005. ACM.
- [13] G. J. Myers. *The Art of Software Testing*. John Wiley and Sons, second edition, 2004. Revised and updated by T. Badgett and T. M. Thomas with C. Sandler.
- [14] D. Slutz. Massive stochastic testing of SQL. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB 1998)*, pages 618–622, 1998.
- [15] R. A. Thayer, M. Lipow, and E. C. Nelson. *Software Reliability*. North-Holland, 1978.
- [16] T. Yoshikawa, K. Shimura, and T. Ozawa. Random program generator for Java JIT compiler test system. In *Proceedings of the 3rd International Conference on Quality Software (QSIC 2003)*, pages 20–24, 2003.
- [17] H. Zhu, P. A. V. Hall, and J. H. R. May. Software unit test coverage and adequacy. *ACM computing surveys*, 29(4):366–427, 1997.

# A Novel Method of Mutation Clustering Based on Domain Analysis\*

Changbin Ji<sup>1,2</sup>, Zhenyu Chen<sup>1,2</sup>, Baowen Xu<sup>1</sup>, Zhihong Zhao<sup>1,2</sup>

<sup>1</sup>National Key Laboratory for Novel Software Technology, Nanjing University, P.R. China

<sup>2</sup>Software Institute, Nanjing University, P.R. China

Corresponding Author: zychen@software.nju.edu.cn

## Abstract

*Mutation testing is an effective but expensive technique. There exists much improvement to help mutation testing become a wide-used technique. The main objective is to reduce the number of mutants and the reduced test sets can still approximate the adequacy. Recently a new method, called mutation clustering, is proposed to integrate data clustering and mutation analysis, such that both mutant sets and test sets can be reduced dramatically. This paper provides domain analysis to cluster mutants statically before test case generation. And then the simplified mutant set is used to generate a test set. We hypothesize that such a test set can kill the original mutant set approximately. The cost of mutation testing will further decrease, because only a small test set needs to be generated and executed. A case study shows an encouraging result to support our hypothesis.*

## 1. Introduction

Mutation testing is testified to be an effective fault detective method [1]. Initially it introduces some minor changes of the original program to produce mutants. A test case is said to kill a mutant if it can detect the corresponding change of program. A test set has a mutation score as a label to illustrate its fault detection ability.

Mutation analysis has many advantages in fault-based testing [2]. However it is still on the way to be a commercial testing technique because of some industrial reasons [3]. One problem of mutation testing is computational expense. Even a program with 30 lines of codes may generate hundreds of mutants. It will spend considerable time and resources to execute so many mutants. On the other hand, there exist some syntactically different but semantically same mutants, which are called equivalent mutants. No test case can kill equivalent mutants so that they must be removed from the original mutant set. Equivalent mutants are

usually distinguished by hand so far. It will cost too much human resources in some cases.

Many techniques have been proposed to reduce the computational expense of mutation testing [3]. There are mainly three kinds of strategies, including “do smarter”, “do faster” and “do fewer”, to improve the performance of mutation testing. A “do smarter” approach pays attention to compare the intermediate mutated states. A “do faster” approach reduces the cost of running mutants in the same environment [4]. A “do fewer” method is more straightforward than the other two. A “do fewer” method tries to execute fewer mutants against fewer test cases [3].

In mutation testing, the mutants are generated based on some mutation operators. The number of mutants can be reduced by selecting some key mutation operators. A. J. Offutt et al. proved that mutation testing will keep its testing strength with fewer mutation operators [3]. Wong et al. introduced selective mutation criterion to choose mutation operators according to their fault detection abilities [5, 6]. The work in [7] further indicated that ABS, AOR, LCR, ROR and UOI were five necessary mutation operators, which provided the same coverage as the non-selective mutation. Random sampling can also avoid executing large number of mutants. It randomly selects some mutants from the initial mutant set and is testified to be effective in some cases [8]. Chen et al. used fault class hierarchy to skip some mutation operators in Boolean specification-based testing [9]. Moreover, some graph contraction strategies were provided to merge some mutants from a same mutation operator, such that the test set is reduced [10].

S. Hussain et al. introduced mutation clustering which could lower the numbers of mutants and test cases at the same time [13]. They firstly run all mutants against all test cases. The execution information was saved as the measurement for every mutant. And then the distances of mutants were calculated and some clustering algorithms were used to

---

\* The work described in this article was partially supported by the National Natural Science Foundation of China (90818027, 60773104, 60803007, 60803008, 60873050, and 60873049).

assemble the similar mutants. However, the preprocessing of mutation clustering was still operationally expensive, because all test cases should be generated and executed before clustering. In this paper, we present a domain analysis method to measure mutants without the execution information of test cases. The measurement information is obtained statically. Hence the cost of mutation testing decreases to make it more applicable.

This paper is organized as follows: Section 2 describes relative work of mutation clustering. In Section 3, some strategies of domain analysis are proposed to prepare for mutation clustering. Experimental results and analysis is shown in Section 4. Some directions in Section 5 will be done in the future.

## 2. Related Work

Data clustering is usually used in pattern-analysis, decision making, knowledge discovery and machine learning situations [11]. It is called a non-supervised method to collect similar data elements. Using a suitable clustering algorithm [12], the data items which satisfy the similarity requirements will be clustered into a same category. The set of clusters acts as a data set but has a smaller scale. S. Hussain et al. apply clustering techniques to minimize the quantity of mutants [13]. A mutant is represented by the test cases that kill it and the test cases that do not kill it. The discrimination between mutants is measured by Hamming distance. Then they apply K-means and agglomerative clustering algorithm to turn the large number of mutants into fewer clusters [14]. Then these clusters reduce the test cases under the greedy algorithm. All mutants must be executed against all test cases before clustering. We propose a more effective way to divide the mutants before test case generation. We adopt another method which can be settled statically to mark every mutant for clustering.

CBT and DDR are two methods that can generate test cases automatically in mutation testing. Both of them focus on the domains of variables. CBT takes the algebraic expressions as constraints [15]. Constraints are reduced dimension by dimension and the test cases will be generated to approximate relative adequacy. DDR takes the initial set for each input [16]. When executing through the control flow of the program, the sets of values are modified dynamically. DDR resolves several shortcomings which CBT suffers. These two methods use values of variables to generate test cases. Hence we hope the scales of variables can also be used to separate mutants.

The application of mutation operators firstly produces some mutants. According to domain analysis,

one value set corresponds to one mutant. A clustering methodology is employed here to cluster the mutants. We select one mutant from each cluster randomly. These mutants compose the next generation to choose. The next section will introduce the domain analysis by which we reduce the domains of variables.

## 3. Mutation Clustering

We intend to apply a static method to measure all mutants. The static method does not need to execute mutants and provides improvement of mutation clustering.

### 3.1. Measure every mutant statically

All inputs in one program are picked out. For every mutant, we use some symbolic execution rules which will be depicted in subsection 3.3 to reduce the domains of all variables [17]. The reduced variables are taken as the representation of a mutant. This method saves execution cost before mutation clustering. Basically we apply the static control flow analysis which has practical cost [18].

### 3.2. Optional module

```
BEGIN
  Module 1:
  if(condition 1)
    return -1;
  Module 2:
  if(condition 2)
    return -1;
  Optional Module 3:
  if(condition 3)
    define or refer some variables;
  Module 4:
  if(condition 4)
    if(condition 5)
      mutated statement
    return 0;
END
```

**Figure 1.** Module structure of a simple program

Figure 1 depicts a simple program. The program has three possible exits under three varying conditions. We take the clauses which are not affected by mutation operators and contain no exit statement as a normal module. For example, one mutant is generated from one statement in module 4. The module 3 is optional and the conditions in module 3 will be dismissed. We try to find a near optimal method and the conditions to analyze need not to be continuous [19]. Identification of modules in the program is used to prepare for the further analysis. The thinking of reused software information accelerates determining the location of every mutant [20].



### 3.3. Reduction conditions for domains

The conditions in the non-optional modules are under consideration. The domains will be reduced using DDR by the combination conditions. There are three types of conditions to combine:

1. Exit-against conditions: If one mutant works, the execution must satisfy the basic condition that the program would not exit before the mutation point. We choose mutant path which would avoid program exiting. As it is shown in Figure 1, the exit-against condition is  $(\neg condition1 \wedge \neg condition2)$ .

2. Nested conditions: In one module, there may be some nested conditions. All correlative conditions up the mutation point will be satisfied. In this example, the nested condition is  $(condition4 \wedge condition5)$ .

3. Mutated conditions: Mutants which derive from the same statement have the equivalent exit-against conditions and nested conditions. The altered condition is treated as a mutated condition so that we can tell differences between mutants generated from the same statement.

The domains of inputs are bounded dynamically when the inputs meet the conditional constraints above. The limitation is that numeric conditions in the mutation path but not all conditions are collected.

### 3.4. Clustering method

This subsection will introduce how the mutants are clustered.

We use hamming distance to calculate the distance between mutants. Table 1 shows how Hamming distance works in our analysis. A, B, C and return value represent M1 and M2. If the domains differ, 1 is brought in. Otherwise, the distance is 0. The sum of 1 is counted as the distance of different mutants. In Table 1, the distance between M1 and M2 is 3.

**Table 1.** Calculate the distance between two mutants

	A	B	C	Return Value
M1	(-30,30)	(-30,15)	(-15,0)	1
M2	(-30,30)	(-30,30)	(-30,30)	-1
Dis.	0	1	1	1

We apply a procedure like K-means algorithm to cluster mutants. Firstly we randomly choose  $k$  mutants as  $k$  clusters. In this paper  $k$  is the half of total number. Then we evaluate the distance from the remaining mutants to these  $k$  clusters. One mutant joins the cluster whose distance is less than the threshold value which is half of the max distance. K-means algorithm uses the means of elements in clusters. Due to the specific of the value of domain, it is difficult to calculate the means of two or several domains directly. We arbitrarily choose one mutant to represent the centroid of one cluster.

When clustering procedure is done, the  $k$  centroids form the next generation which will be clustered again.

## 4. Experimental Result and Analysis

The aim of this section is to evaluate whether the values of domains can help distinguish the mutants instead of the execution information. At first, we apply muJava<sup>1</sup> to generate mutants  $M_F$  for a program P. Some mutation operators in class are dismissed according to the feature of the case program [21]. Then a test set  $T_F$  is designed for  $M_F$  and evaluated with the mutation score  $MS_F$ . We find all inputs in P including return values and use the same method as DDR to reduce the domains of inputs dynamically by mutated conditions.  $K$  mutants are selected randomly. We cluster the mutants whose distances are less than the threshold value. One mutant is selected randomly to represent one cluster and these mutants form the new mutant collection  $M_N$ . Test cases killing no mutant in  $M_N$  are removed from  $T_F$ . The remaining test cases compose new test set  $T_N$ .  $T_N$  is evaluated with  $MS_N$  to kill the  $M_F$ . The efficiency of  $T_F$  and  $T_N$  is compared.  $M_N$  can be clustered again.

In this case study, we take the triangle program as the study object. Table 2 provides the results of mutation clustering. We manually generate 21 test cases for M1 which is produced by muJava. M1 are divided into 73 clusters. 73 centroids of M1 constitute M2. We remove the test cases that can not kill one mutant in M2. The test cases are reduced from 21 to 16. 16 test cases are used to kill M1. Significantly, the number of killed mutants is reduced only by 4, while the total number of mutants is saved by 50%. Obviously the computation cost of mutation testing falls. One more point worth mentioning is that M3 has only 25% mutants of M1 and 13 test cases filtered by M3 kill 94% mutants.

**Table 2.** Results of mutation clustering

1 Number of test cases:21			
	Mutants	Killed mutants	MS
M1	147	137	93%
M2	73	68	93%
M3	36	35	97%
M4	18	18	100%
2 Number of test cases:16 filtered by M2			
	Mutants	Killed mutants	MS
M1	147	133	90%
M2	73	68	93%
3 Number of test cases:13 filtered by M3			
	Mutants	Killed mutants	MS

<sup>1</sup> <http://cs.gmu.edu/~offutt/mujava/>

M1	147	130	88%
M2	73	66	90%
M3	36	35	97%

4 Number of test case:9 filtered by M4			
	Mutants	Killed mutants	MS
M1	147	117	79%
M2	73	60	82%
M3	36	31	86%
M4	18	18	100%

The number of test cases is reduced by fewer mutants. Whereas, the reduced test set is still as strong as the original test set. The descending of M2 and M3 is acceptable. The positive results testify our hypothesis that values of domains can be used to cluster mutants instead of the execution information. Clustering mutants provides an optimal solution, and it makes mutation testing to execute fewer mutants.

## 5. Future Work

One challenge of our method is how to rationally determine the means of domains as the centroid of a cluster. Another challenge is that this approach is not appropriate for inter-procedural program and will be applied to unit testing. We will envision a fully automatic system to implement the whole procedure of mutation clustering. Then we can apply our analysis on other bigger scale programs. We hope it can help reduce the operational cost and make the mutation testing to be more practical.

## 6. References

[1] A. P. Mathur and W. E. Wong. An Empirical Comparison of Data Flow and Mutation-Based Test Adequacy Criteria. *Software Testing, Verification & Reliability*. 1994, 4(1):9-31.

[2] J. H. Andrews, L. C. Brand and Y. Labiche. Is mutation an appropriate tool for testing experiments?. In *Proceedings of the 27th International Conference on Software Engineering*. 2005, pp.402-411.

[3] A. J. Offutt and R. H. Untch. Mutation 2000: Uniting the Orthogonal. *Kluwer International Series on Advances in Database Systems, Mutation testing for the new century*. 2001, pp.34-44.

[4] R. Untch, A. J. Offutt and M. J. Harrold. Mutation analysis using program schemata. In *Proceedings of the 1993 International Symposium on Software Testing, and analysis*. 1993, pp.139-148.

[5] W. E. Wong, M. E. Delamaro, J. C. Maldonado, and A. P. Mathur. Constrained mutation in C programs. In *Proceedings of VIII Symposium on Software Engineering*. 1994, pp.439-452.

[6] A. J. Offutt, G. Rothermel and C. Zapf. An experimental evaluation of selective mutation. In *Proceedings of the Fifteenth International Conference on Software Engineering*. 1993, pp.100-107.

[7] A. J. Offutt, G. Rothermel and C. Zapf. An experimental determination of sufficient mutation operators. *ACM Transactions on Software Engineering and Methodology*. 1996, 5(2):99-118.

[8] A. P. Mathur and W. E. Wong. Reducing the cost of mutation testing: an empirical study. *Journal of Systems and Software*, 1995, 31(3):185-196.

[9] Zhenyu Chen, Baowen Xu, Xiaofang Zhang and Changhai Nie. A novel approach for test suite reduction based on requirement relation contraction. In *Proceedings of the ACM symposium on Applied Computing*. 2008, pp.390-394.

[10] Zhenyu Chen, Baowen Xu and Changhai Nie. A detectability analysis of fault classes for Boolean specifications. In *Proceedings of the ACM symposium on Applied Computing*. 2008, pp.826-830.

[11] A. K. Jain, M. N. Murty, P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*. 1999, 31(3):264-323.

[12] G. Fung. A Comprehensive Overview of Basic Clustering Algorithms. 2001.

[13] S. Hussain, M. Harman. Mutation Clustering. Master Thesis, King's College London, UK, 2008.

[14] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman and A. Y. Wu. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002, 24(7):881-892.

[15] R. A. DeMillo and A. J. Offutt. Constraint-based automatic test data generation. *IEEE Transaction on Software Engineering*. 1991, 17(9):900-910.

[16] A. J. Offutt, Jin Z and Pan J. The dynamic domain reduction procedure for test data generation. *Software: Practice and Experience*. 1999, 29(2):167-193.

[17] Corina S. Pasareanu and Willem Visser. Verification of Java Programs Using Symbolic Execution and Invariant Generation. 2004, LNCS 2989:0302-9743.

[18] A. Rountev, O. Volgin and M. Reddoch. Static control-flow analysis for reverse engineering of UML sequence diagrams. In *Proceedings of the 6th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering table of contents*. 2005, pp.96-102.

[19] M. Harman. Search Based Software Engineering. 2006, LNCS 3994: 0302-9743.

[20] R. Prieto-Díaz. Domain analysis: an introduction. *ACM SIGSOFT Software Engineering Notes archive*. 1990, 15(2):47-54.

[21] Y. S. Ma, J. Offutt and Y. R. Kwon. MuJava: a mutation system for java. In *Proceedings of the 28th International Conference on Software Engineering*. 2006, pp. 827-830.

# Using a mining frequency patterns model to automate passive testing of real-time systems \*

César Andrés, Mercedes G. Merayo, Manuel Núñez  
Departamento Sistemas Informáticos y Computación  
Universidad Complutense de Madrid  
{c.andres,mgmerayo}@fdi.ucm.es, mn@sip.ucm.es

## Abstract

*Testing is one of the most widely used techniques to increase the confidence on the correctness of complex software systems. In this paper we extend our previous work on passive testing with invariants to incorporate (probabilistic) knowledge obtained from users of the system under test. In order to apply our technique, we need to obtain a set of invariants compiling the relevant properties of the system under test, and this is a difficult task. First, we present an algorithm to extract invariants from the specification without assuming any additional condition. Since the number of obtained invariants is huge we study an alternative. Based on the idea that an invariant is better than another one if it can be checked more times in the same log, we present an adaptation of the previous algorithm in order to sort sets of representative invariants.*

## 1 Introduction

With the growing significance of software systems, techniques that assist in the production of reliable software are becoming increasingly important within software engineering. The complexity of current software systems requires the application of sound techniques. Among these techniques, testing [13] is the most widely used in industrial environments. Unfortunately, testing is still a mainly manual activity, prone to errors that can mask real errors of the tested system. Traditionally, formal methods and testing have been seen as rivals. Thus, there was very little interaction between the two communities. In recent years, however, these approaches are seen as complementary [9, 8].

\*Research supported by the Spanish MEC project WEST/FAST (TIN2006-15578-C02-01), the MATES project (CCG08-UCM/TIC-4124) and by the UCM-BSCH programme to fund research groups (GR58/08 - group number 910606).

In most cases, testing is based on the ability of a tester that stimulates the implementation under test (IUT) and checks the correction of the answers provided by the implementation. However, in some situations this activity becomes difficult and even impossible to perform. For example, this is the case if the tester is not provided with a direct interface to interact with the IUT. Another conflictive situation appears when the implementation is built from components that are running in their environment and cannot be shutdown or interrupted for a long period of time. In these situations, there is a particular interest in using other types of validation techniques such as *passive testing*. Therefore, the main difference between active and passive testing is that in active testing testers can interact, by providing inputs, with the IUT and observe the obtained result.

Passive testing usually consists in recording the trace produced by the IUT and trying to find a *fault* by comparing this trace with the specification [10, 14, 12, 5]. A new methodology to perform passive testing was presented in [6, 4]. The main novelty is that a set of *invariants* is used to represent the most relevant properties of the specification. An invariant can be seen as a restriction over the traces allowed to the IUT. We introduced in previous work [3] the possibility of adding time constraints as properties that traces extracted from the IUT must hold.

Normally, there are several ways to obtain a set of invariants. The first one is that testers provide a set of representative invariants. Then, the correctness of these invariants, with respect to the specification, has to be checked. The main drawback of this approach is that we still have to rely on the manual work of the tester. If we do not have a complete formal specification, a variant of this approach is to assume that the set of invariants provided by the testers are correct by definition. In this paper we consider an alternative approach: We automatically derive invariants from the

specification. First, we consider an adaptation of the algorithms presented in [6]. The problem with this first attempt is that the number of invariants that we extract from the specification is huge and we do not have a criterion to decide which ones are *better*. We propose an alternative algorithm to use knowledge extracted from *standard* users of the system, so that invariants can be sorted according to their *quality* to fit the behavior of those users.

The rest of the paper is structured as follows. In Section 2 we present our passive testing framework of timed systems. In Section 3, containing the bulk of the paper, we present two algorithms for extracting a set of correct invariants from the specification. We conclude with Section 4, where we present the conclusions and some lines of future work.

## 2 A formal framework for passive testing of timed systems

In this section we introduce our framework to specify and (passively) test timed systems. We extend the well-known Finite State Machines model by adding time information concerning the amount of time that the system needs to perform transitions.

**Definition 1** A *Timed Finite State Machine (TFSM)* is a tuple  $(\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$ , where  $\mathcal{S}$  is a finite set of states,  $s_0 \in \mathcal{S}$  is the initial state,  $\mathcal{I}$  and  $\mathcal{O}$ , with  $\mathcal{I} \cap \mathcal{O} = \emptyset$ , are the finite sets of *input* and *output* actions, respectively, and  $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{I} \times \mathcal{O} \times \mathbb{R}_+ \times \mathcal{S}$  is the set of transitions.

A TFSM  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  is *deterministic* if for all state  $s$  and input  $i$  there exists at most one transition  $(s, i, o, t, s')$ . We say that  $M$  is *input-enabled* if for all state  $s \in \mathcal{S}$  and  $i \in \mathcal{I}$  there exists a transition  $(s, i, o, t, s')$ .  $\square$

Let us consider the TFSM depicted in Figure 1 and the transition  $(s_1, a, x, 3, s_2)$ . Intuitively, if the machine is at state  $s_1$  and it receives the input  $a$ , then it will produce the output  $x$  after 3 time units. We usually write  $s \xrightarrow{i/o}_t s'$  as a shorthand of  $(s, i, o, t, s') \in \mathcal{T}$ . As usual in formal testing approaches, we assume that both specifications and implementations can be represented by the same formalism. In our case, the formalism is the set of deterministic input-enabled TFSMs.

Next we introduce the notion of *trace* of a system. A trace captures the behaviour of an implementation. Traces collect the outputs obtained after sending some inputs to the implementation and the amount of time between each input/output pair. The classical notion of trace, which does not include any time information, is called a *non-timed trace*.

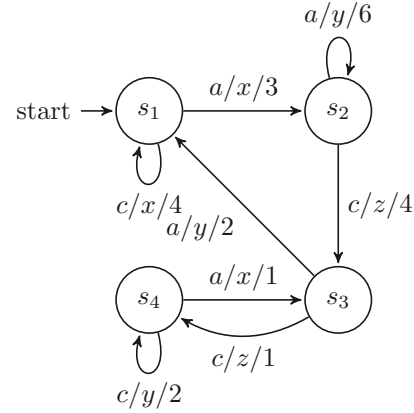


Figure 1. Example of TFSM.

**Definition 2** Let  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  be a TFSM. We say that  $\theta = \langle i_1/o_1/t_1, i_2/o_2/t_2, \dots, i_n/o_n/t_n \rangle$  is a *timed trace*, or simply *trace*, of  $M$  if there is a sequence of transitions such that

$$s_1 \xrightarrow{i_1/o_1}_{t_1} s_2 \xrightarrow{i_2/o_2}_{t_2} s_3 \dots s_{n-1} \xrightarrow{i_n/o_n}_{t_n} s_n$$

We denote the empty trace by  $\langle \rangle$  and by  $\text{tr}(M)$  the set of all traces of  $M$ .

If  $\theta = \langle i_1/o_1/t_1, \dots, i_n/o_n/t_n \rangle$  is a timed trace of  $M$ , then the sequence  $\omega = \langle i_1/o_1, \dots, i_n/o_n \rangle$  is a *non-timed trace* of  $M$ . We denote by  $\text{nttr}(M)$  the set of all non-timed traces of  $M$ .  $\square$

For example, if we consider the machine  $M$  depicted in Figure 1, we have that  $\langle c/x/4, c/x/4, a/x/3, c/z/4, c/z/1, a/x/1 \rangle$  is a timed trace of  $M$ . If we remove time information, we obtain *non-timed* traces. For instance,  $\langle c/x, c/x, a/x, c/z, c/z, a/x \rangle$  is the non-timed trace associated with the previous timed trace.

Our passive testing approach is similar to other methodologies since the basic idea consists in recording traces from the IUT to detect unexpected behaviours. The main novelty is that a set of *invariants* is used to represent the most relevant properties of the specification. Intuitively, an invariant expresses the fact that each time the IUT performs a given sequence of actions, then it must exhibit a behaviour reflected in the invariant. In order to express traces in a concise way, we will use the wild-card characters  $?$  and  $\star$ . The wild-card  $?$  represents any value in the sets of inputs or outputs, while  $\star$  represents a sequence of input/output pairs.

**Definition 3** We say that  $\hat{p} = [p_1, p_2]$  is a *time interval* if  $p_1 \in \mathbb{R}_+$ ,  $p_2 \in \mathbb{R}_+ \cup \{\infty\}$ , and  $p_1 \leq p_2$ .

We assume that for all  $t \in \mathbb{R}_+$  we have  $t < \infty$  and  $t + \infty = \infty$ . We consider that  $\mathcal{IR}$  denotes the set of time intervals.

Let  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  be a TFSM. We say that the sequence  $\varphi$  is an *invariant* for  $M$  if the following two conditions hold:

1.  $\varphi$  is defined according to the following EBNF:

$$\begin{aligned}\varphi & ::= a/z/\hat{p}, \varphi \mid \star/\hat{p}, \varphi' \mid i \mapsto O/\hat{p} \triangleright \hat{t} \\ \varphi' & ::= i/z/\hat{p}, \varphi \mid i \mapsto O/\hat{p} \triangleright \hat{t}\end{aligned}$$

In this expression we consider  $\hat{p}, \hat{t} \in \mathcal{IR}$ ,  $i \in \mathcal{I}$ ,  $a \in \mathcal{I} \cup \{?\}$ ,  $z \in \mathcal{O} \cup \{?\}$ , and  $O \subseteq \mathcal{O}$ .

2.  $\varphi$  is *correct* with respect to  $M$  (formally defined in [3]).

□

Even though, our machines present time information expressed as fix amounts of time, time conditions established in invariants are given by intervals. This fact is due to consider that it can be admissible that the execution of a task sometimes lasts more time than expected: If most of the times the task is performed on time, a small number of delays can be tolerated. Concerning the notion of correctness, the idea is that an invariant is correct with respect to a machine  $M$  if  $M$  cannot perform a sequence of transitions that would contradict what the invariant expresses.

Intuitively, the previous EBNF expresses that an invariant is either a sequence of symbols where each component, but the last one, is either an expression  $a/z/\hat{p}$ , with  $a$  being an input action or the wild-card character  $?$ ,  $z$  being an output action or  $?$ , and  $\hat{p}$  being an interval, or an expression  $\star/\hat{p}$ . There are two restrictions to this rule. First, an invariant cannot contain two consecutive expressions  $\star/\hat{p}_1$  and  $\star/\hat{p}_2$ . In the case that such situation was needed to represent a property, the tester could simulate it by means of the expression  $\star/(\hat{p}_1 + \hat{p}_2)$ . The second restriction is that an invariant cannot present a component of the form  $\star/\hat{p}$  followed by an expression beginning with the wild-card character  $?$ , that is, the input of the next component must be a *real* input action  $i \in \mathcal{I}$ . In fact,  $\star$  represents any sequence of inputs/outputs pairs such that the input is not equal to  $i$ .

The last component, corresponding to the expression  $i \mapsto O/\hat{p} \triangleright \hat{t}$  is an input action followed by a set of output actions and two timed restrictions, denoted by means of two intervals  $\hat{p}$  and  $\hat{t}$ . The former is associated to the last expression of the sequence. The latter is related to the sum of times values associated to all input/output pairs performed before. For example, the

meaning of an invariant as  $i/o/\hat{p}, \star/\hat{p}_\star, i' \mapsto O/\hat{p}' \triangleright \hat{t}$  is that if we observe the transition  $i/o$  in a time belonging to the interval  $\hat{p}$ , then the first occurrence of the input symbol  $i'$  after a lapse of time belonging to the interval  $\hat{p}_\star$ , must be followed by an output belonging to the set  $O$ . The interval  $\hat{t}$  makes reference to the total time that the system must spend to perform the whole trace. Next we introduce some examples in order to present how invariants work.

**Example 1** Let us consider the TFSM presented in Figure 1. One of the most simple invariants that we can define within our framework follows the scheme  $a \mapsto \{x, y\}/[1, 6] \triangleright [1, 6]$ . The idea is that each occurrence of the symbol  $a$  is followed by either the output symbol  $x$  or  $y$  and this transition is performed between 1 and 6 time units.

We can specify a more complex property by taking into account that we are interested in observing the output  $x$  after the input  $a$  when the pair  $c/x$  was previously observed. In addition, we include time intervals corresponding to the amount of time the system takes for each of the transitions and to the total time it spends in the whole trace. We could express this property by means of the invariant  $c/x/[3, 5], \star/[0, 5], a \mapsto \{x\}/[2, 4] \triangleright [4, 13]$ . An observed trace will be correct with respect to this invariant if each time that we find a (sub)sequence starting with the  $c/x$  pair, performed in an amount of time belonging to the interval  $[3, 5]$ , if there is an occurrence of the input  $a$  before 5 time units pass, then it must be paired with the output symbol  $x$  and the lapse of time between  $a$  and  $c$  must belong to the interval  $[2, 4]$ . In addition, the whole sequence must take a time belonging to the interval  $[4, 13]$ . □

### 3 Two approaches for the extraction of invariants from specifications

Normally, invariants should be supplied by the expert/tester. Then, the first step is to check that the invariants are correct with respect to the specification. Another approach consists in extracting invariants from the specification, so that their correctness is ensured. In this section we present two algorithms to extract invariants from specifications. The first algorithm is based on the adaptation to our framework of the algorithms given in [6]. The main drawback of applying this approach is that, in general, we have a huge set of invariants. The second algorithm, uses extra information about the users of the system and improves the first algorithm in two features. The first one is that the set of invariants is smaller. The second one is that

---

```

in :  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T}) : \text{TFSM}$ 
out:  $\phi$  : Set of correct invariants
1  $\Omega = \text{nttr}(M)$ ;
2  $\phi = \emptyset$ ;
3 while ( $\Omega \neq \emptyset$ ) do
4   | Choose  $\omega \in \Omega$ ;
5   |  $\Omega = \Omega \setminus \{\omega\}$ ;
6   |  $\phi = \phi \cup \text{generateT}(M, \omega)$ ;
7 end
8 return  $\phi$ ;

```

---

**Figure 2. Algorithm for generating invariants from a TFSM adapting algorithms in [6].**

---

we can establish a minimum *significance degree* value for the set of invariants.

The adaptation to our framework of the algorithms given in [6] is presented in Figure 2. The next auxiliary functions are used in the algorithm.

**Definition 4** Let  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  be a TFSM and  $\omega = \langle i_1/o_1, \dots, i_n/o_n \rangle$  be a non-timed trace of  $M$ . The function  $\text{outputs}(M, \omega)$  computes the set of all possible outputs associated with the last input of  $\omega$ .

$$\text{outputs}(M, \omega) = \{o \mid \langle i_1/o_1, \dots, i_n/o \rangle \in \text{nttr}(M)\}$$

The functions  $\text{mT}(\omega, j, O)$  and  $\text{MT}(\omega, j, O)$  compute the minimum, respectively maximum, time value associated to the pair  $i_j/o_j$  in all the timed traces of  $M$  corresponding to  $\omega$ , except the last output that must belong to  $O$ , that is,

$$\text{mT}(\omega, j, O) = \min \{t_j \mid \langle i_1/o_1/t_1, \dots, i_n/o/t_n \rangle \in \text{tr}(M) \wedge o \in O\}$$

$$\text{MT}(\omega, j, O) = \max \{t_j \mid \langle i_1/o_1/t_1, \dots, i_n/o/t_n \rangle \in \text{tr}(M) \wedge o \in O\}$$

The function  $\text{generateT}(M, \omega)$  computes a set of correct invariants with respect to  $M$  corresponding to the sequence  $\omega$ .

$$\left\{ \varphi \mid \begin{array}{l} \varphi = i_1/o_1/\hat{p}_1, \dots, i_n \mapsto O/\hat{p}_n \triangleright \hat{q} \wedge \\ O = \text{outputs}(M, \omega) \wedge \\ \forall 1 \leq k \leq n : \hat{p}_k = [\text{mT}(\omega, k, O), \text{MT}(\omega, k, O)] \wedge \\ \hat{q} = \sum \hat{p}_k \end{array} \right\}$$

□

Next, we briefly describe this algorithm. The input parameter is the specification of the system, while the output is a set of correct invariants generated from it. The algorithm applies the function  $\text{generateT}$  to all the non-timed traces  $\Omega$  of the specification in order to

produce a set of correct invariants  $\phi$ . The drawback of this proposal is that the number of possible invariants is really huge, usually infinite, and most of them have low significance. Let us note that this set is usually infinite due to the fact that we compute all non-timed traces of the specification.

In order to reduce the number of invariants we propose an algorithm that uses extra information obtained by the application of data mining techniques to a set of data recorded from the interaction between different users and the IUT. First, a selection of data is made and preprocessed in order to check that there does not exist any incongruence, that is, the set of data represents traces of users that have been observed during their interaction with the IUT. We will focus on extracting sets of *relevant behaviors* from this set of data, that is, those sequences of interactions of the users with the system that appear more frequently. In order to determine these behaviors we use the usual techniques for obtaining frequent patterns from a database (see, for example, [1, 11]). The task of discovering the frequency of each behaviour in the database is performed by means of the applications of any of the existing tools. In our approach we use [7].

In order to determine the knowledge obtained from the data, we formally define a *frequency threshold* relative to a set of sequences. The frequency threshold represents the degree of relative frequency that a set of behaviours has with respect to the global set of behaviours [15]. Let us note that in order to define the user behaviour, we abstract outputs and time values since they are not relevant for this phase.

**Definition 5** A *user behaviour* is a sequence of input actions  $\alpha = \langle i_1, i_2, \dots, i_n \rangle$  where  $n \geq 0$ . A *database*  $\mathcal{DB}$  is a multiset of user behaviours.

If  $\alpha'$  is a subtrace of  $\alpha$  then we denote it by  $\alpha' \leq \alpha$ . The set of all prefixes of  $\alpha$  is denoted by  $\text{tu}(\alpha)$ .

We denote by  $\text{prob}(\alpha, \mathcal{DB})$  the probability of finding the user behaviour  $\alpha$  in the database  $\mathcal{DB}$ .

The *prefix-closed frequency* of a set of input sequences  $\Sigma$  over the database  $\mathcal{DB}$  is defined as:

$$c_{\mathcal{DB}}(\Sigma) = \sum_{\alpha \in \bigcup_{\alpha' \in \Sigma} \text{tu}(\alpha')} \text{prob}(\alpha, \mathcal{DB})$$

□

We assume that the world is *regular*. Therefore, if we have a big number of user behaviours in the database, we are able to consider that we have an acceptable percentage of the amount of usual interactions with the system. Let us note that the idea of being provided with a knowledge base which contains a representation

of an expertise in the domain, that is our database, is not new; it is a usual assumption considered in a expert system.

In Figure 3 we adapt the previous algorithm presented in Figure 2 to include the extra information that we have obtained by using our data mining process. The goal of this algorithm is still to obtain a set of correct invariants from the specification. The problem that we had in the previous algorithm, regarding to the high number of invariants that we generated, is solved by focusing only on the invariants that represent a *normal* behaviour of a user interacting with the IUT. Let us remember that we abstract the outputs and the time values.

Next we define a set of auxiliary functions which are used in the algorithm.

**Definition 6** Let  $\mathcal{DB}$  be a database. The function  $\text{msTOS}$  computes the set of user behaviours associated to a database, that is,

$$\text{msTOS}(\mathcal{DB}) = \{\alpha \mid \alpha \in \mathcal{DB}\}$$

The function  $\text{moreR}$  computes the most representative sequence of input actions of a database  $\mathcal{DB}$  that does not belong to a set of input sequences  $\Sigma$ , that is,

$$\text{moreR}(\mathcal{DB}, \Sigma) = \begin{cases} \alpha & \text{if } \exists \alpha : \alpha \in \text{msTOS}(\mathcal{DB}) \setminus \Sigma \wedge \\ & \forall \alpha' : \alpha' \in \text{msTOS}(\mathcal{DB}) \setminus \Sigma \wedge \\ & c_{\mathcal{DB}}(\{\alpha\}) \geq c_{\mathcal{DB}}(\{\alpha'\}) \\ \langle \rangle & \text{otherwise} \end{cases}$$

Let  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  be a TFMSM, and  $\alpha = \langle i_1, \dots, i_n \rangle$  be a single sequence of input actions. The function  $\text{generateT}'(M, \alpha)$  computes a set of correct invariants with respect to  $M$  corresponding to the inputs sequence  $\alpha$ , that is,

$$\left\{ \varphi \mid \begin{array}{l} \varphi = h_1, \dots, h_{n-1}, i_n \mapsto \{O\} / \hat{p}_n \triangleright \hat{q} \wedge \\ h_1 = i_1 / o_1 / \hat{p}_1 \wedge \dots \wedge h_{n-1} = i_{n-1} / o_{n-1} / \hat{p}_{n-1} \wedge \\ \exists \omega = \langle i_1 / o_1, \dots, i_n / o \rangle \in \text{nttr}(M) \wedge \\ O = \text{outputs}(M, \omega) \wedge \\ \forall 1 \leq k \leq n : \hat{p}_k = [\text{mT}(\omega, k, O), \text{MT}(\omega, k, O)] \wedge \\ \hat{q} = \sum \hat{p}_k \end{array} \right\}$$

□

Next, we briefly describe the main steps of this algorithm. As input parameters of the algorithm we have the specification, modelled using a TFMSM, the database  $\mathcal{DB}$  containing the stored sequences of interactions between users and the IUT and, the significance degree over the database that it is required. The results of our algorithm is a set of correct invariants  $\phi$ .

---

```

in :  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  : TFMSM
       $\mathcal{DB}$ : Database.
       $\mathcal{C} : \mathbb{R}_+$  Grade of relevance.
out:  $\phi$ : Set of correct invariants.
1  $\Sigma = \emptyset$ ;  $\phi = \emptyset$ ;
2 while ( $c_{\mathcal{DB}}(\Sigma) < \mathcal{C}$ )  $\wedge$  ( $\text{msTOS}(\mathcal{DB}) \setminus \Sigma \neq \emptyset$ ) do
3   |  $\alpha = \text{moreR}(\mathcal{DB}, \Sigma)$ ;
4   |  $\Sigma = \Sigma \cup \{\alpha\}$ ;
5 end
6 // We will enter this loop only if maximum
7 // coverage is not reached
8 while ( $c_{\mathcal{DB}}(\Sigma) < \mathcal{C}$ ) do
9   | Choose  $i \in \mathcal{I}$ ;  $\Sigma' = \Sigma$ ;  $\Sigma = \emptyset$ ;
10  | while  $\Sigma' \neq \emptyset$  do
11  |   | Choose  $\alpha \in \Sigma'$ ;  $\Sigma' = \Sigma' \setminus \{\alpha\}$ ;
12  |   |  $\Sigma = \Sigma \cup \{\alpha\} \cup \{i \cdot \alpha\}$ ;
13  |   end
14 end
15 // Loop to generate invariants from stored
16 // information
17 while ( $\Sigma \neq \emptyset$ ) do
18  | Choose  $\alpha \in \Sigma$ ;  $\Sigma = \Sigma \setminus \{\alpha\}$ ;
19  |  $\phi = \phi \cup \text{generateT}'(M, \alpha)$ ;
20 end
21 return  $\phi$ ;

```

---

**Figure 3. Algorithm for generating invariants from a TFMSM using statistical information.**

We have two different phases in this algorithm. The first one selects a set of user behaviours  $\Sigma$  which represents at least the grade of relevance established. Initially we choose the most significant sequences of user interactions from the database applying the function  $\text{moreR}$ . If the grade of relevance that we obtain after computing all sequences of user interactions in the database is less than the one required, we extend the traces in  $\Sigma$  until we reach it. The second phase uses the set  $\Sigma$  and generates a set of correct invariants with respect to the provided specification, storing them in  $\phi$  which will be the set that the algorithm returns. The function  $\text{generateT}'$ , given in Definition 6 is used for generating a set of correct invariants given a sequence of inputs. Let us remark that this function is different from the function  $\text{generateT}$  used in the previous algorithm. The new function considers a sequence of inputs, while the previous one considered a sequence of input/output pairs.

When we compare the set of invariants obtained from this second algorithm with respect to the set of invariants obtained from the first algorithm we see that

the cardinal of this new set is always less than or equal to the obtained with respect to the first one.

## 4 Conclusions and Future Work

In this paper we have extended our passive testing framework with two algorithms to generate invariants from the specification. Our invariants can be seen as rules that reflect functional and non-functional aspects of the specification that must be hold by any trace produced by the IUT. As usual, the set of all possible traces from an implementation is infinite and we want to generate a representative set of invariants which reflect the most often interactions between users and implementation. We present a methodology based on data mining from a database which contains the interactions between users and the system, in order to obtain the most frequent interactions sequences. We use this extra information to guide the second algorithm.

As future work we will focus on adapting a formal framework, such as the one presented in [2], to the task of representing a user model which reflects the behaviour of a user. We will use this model to reflect the probability to perform an action and we will adapt the second algorithm provided in this paper with this new formalism.

## References

- [1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *19th ACM Int. Conf. on Management of Data, SIGMOD'93*, pages 207–216. ACM Press, 1993.
- [2] C. Andrés, L. Llana, and I. Rodríguez. Formally comparing user and implementer model-based testing methods. In *4th Workshop on Advances in Model Based Testing, A-MOST'08*, pages 1–10. IEEE Computer Society Press, 2008.
- [3] C. Andrés, M.G. Merayo, and M. Núñez. Passive testing of timed systems. In *6th Int. Symposium on Automated Technology for Verification and Analysis, ATVA'08, LNCS 5311*, pages 418–427. Springer, 2008.
- [4] E. Bayse, A. Cavalli, M. Núñez, and F. Zaïdi. A passive testing approach based on invariants: Application to the WAP. *Computer Networks*, 48(2):247–266, 2005.
- [5] A. Benharref, R. Dssouli, M. Serhani, and R. Glitho. Efficient traces' collection mechanisms for passive testing of web services. *Information & Software Technology*, 51(2):362–374, 2009.
- [6] A. Cavalli, C. Gervy, and S. Prokopenko. New approaches for passive testing using an extended finite state machine specification. *Information and Software Technology*, 45:837–852, 2003.
- [7] E. Frank, M. Hall, G. Holmes, R. Kirkby, and B. Pfahringer. Weka - a machine learning workbench for data mining. In O. Maimon and L. Rokach, editors, *The Data Mining and Knowledge Discovery Handbook*, pages 1305–1314. Springer, 2005.
- [8] R.M. Hierons, K. Bogdanov, J.P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luetzgen, A.J.H Simons, S. Vilkomir, M.R. Woodward, and H. Zedan. Using formal methods to support testing. *ACM Computing Surveys*, 41(2), 2009.
- [9] R.M. Hierons, J.P. Bowen, and M. Harman, editors. *Formal Methods and Testing, LNCS 4949*. Springer, 2008.
- [10] D. Lee, A.N. Netravali, K.K. Sabnani, B. Sugla, and A. John. Passive testing and applications to network management. In *5th IEEE Int. Conf. on Network Protocols, ICNP'97*, pages 113–122. IEEE Computer Society Press, 1997.
- [11] T. Mielikäinen. Frequency-based views to pattern collections. *Discrete Applied Mathematics*, 154(7):1113–1139, 2006.
- [12] R. E. Miller, D. Chen, D. Lee, and R. Hao. Coping with nondeterminism in network protocol testing. In *17th Int. Conf. on Testing of Communicating Systems, TestCom'05, LNCS 3502*, pages 129–145. Springer, 2005.
- [13] G.J. Myers. *The Art of Software Testing*. John Wiley and Sons, 2nd edition, 2004.
- [14] M. Tabourier and A. Cavalli. Passive testing and application to the GSM-MAP protocol. *Information and Software Technology*, 41:813–821, 1999.
- [15] D. Taniar and J. Goh. On mining movement pattern from mobile users. *International Journal of Distributed Sensor Networks*, 3(1):69–86, 2007.



# DeLLIS: a Data Mining Process for Fault Localization

Peggy Cellier, Mireille Ducassé, Sébastien Ferré, Olivier Ridoux  
University of Rennes 1/INSA of Rennes/IRISA \*

## Abstract

*Most dynamic fault localization methods aim at totally ordering program elements from highly suspicious to innocent. This ignores the structure of the program and creates clusters of program elements where the relations between the elements are lost. We propose a data mining process that computes program element clusters and that also shows dependencies between program elements. Experimentations show that our process gives a comparable number of lines to analyze than the best related methods while providing a richer environment for the analysis. We also show that the method scales up by tuning the statistical indicators of the data mining process.*

## 1. Introduction

A large amount of information can be collected from program executions, for instance the executed lines, the values of variables or the values of predicates. We call that information the *events* of the execution, they constitute the *execution trace*. Dynamic fault localization methods that compare execution traces, distinguish two sorts of executions, *passed* and *failed* executions. Passed executions behave according to their specification, failed executions do not. These methods are based on the intuition that, firstly, the events that appear only in the traces of failed executions are more likely to be erroneous, and, secondly, the events that appear only in the traces of passed executions are more likely to be innocent. For instance, the nearest neighbor method, by Renieris and Reiss [11], takes into account only one failed execution. A passed execution “near” to the failed execution is then selected. The difference between the trace of the failed execution and the trace of the passed execution is presented as the suspicious events. Events are the executed lines. For the nearest neighbor method, the lines that appear in the traces of both passed and failed executions are innocent. However, many trace events belong to the traces of both passed and failed executions. The worst

case of that method appears when the closest passed execution executes the same lines as the failed execution; no line is suspicious.

In order to solve that problem, the Tarantula method, by Jones *et al.* [8], considers as suspicious the lines that are executed “often” by failed executions and “seldom” by passed executions. The notion of “often” and “seldom” is measured by a score value. A score, assigned to each line of the program, takes into account the frequencies at which the line appears in the trace of failed and passed executions. Lines which are executed together for all executions have the same score value, for example lines that belong to the same basic block. Unfortunately, lines can have the same score value whereas there is no relation between them. Conversely, let us assume a conditional statement in which a branch always fails and the other branch always succeeds. The condition, the fail branch and the success branch may have very different scores although they are closely related. Thus, completely ordering events by their score may aggregate unrelated events and may separate related events. Nevertheless, most of the fault localization methods [4, 9, 10, 12, 13] try to give a total ordering of the trace events based on their score.

In this paper we describe DeLLIS, a process that takes into account the dependencies between elements of the traces. Thanks to the combination of two data mining techniques, DeLLIS computes all differences between execution traces and, at the same time, gives a partial ordering of those differences. Thus, DeLLIS gives more than a set of independent lines like the other methods, it also highlights the existing relations between them. The data mining details are given in [2]. This article focuses on experimentations, from a software engineering perspective. We first show on the programs of the Siemens suite [5], provided by SIR (Software-artifact Infrastructure Repository<sup>1</sup>) that our method gives a comparable number of lines to analyze while providing a richer environment for the analysis. Then, we show that the method scales up for a program of several thousands of lines by tuning the statistical indicators of the data mining techniques.

In the sequel, Section 2 briefly presents two techniques

\*firstname.lastname@irisa.fr

<sup>1</sup><http://sir.unl.edu/content/sir.html>

	size			sun distance		moons	
	small	medium	large	near	far	with	without
Mercury	x			x			x
Venus	x			x			x
Earth	x			x			
Mars	x			x		x	
Jupiter			x		x	x	
Saturn			x		x	x	
Uranus		x			x	x	
Neptune		x			x	x	

Table 1. Context of planets.

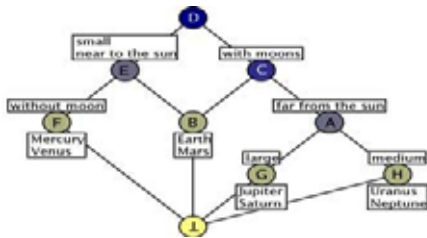


Figure 1. Concept lattice of Table 1

of data mining. Section 3 describes how they are used for fault localization. Section 4 shows the experiments.

## 2. Background Knowledge on Data Mining

We briefly present the two data mining techniques used in the sequel: Association Rules (AR) and Formal Concept Analysis (FCA). AR allows interesting regularities to be found. FCA allows relevant clusters to be computed and partially ordered. The input of both techniques is a *formal context*, i.e. a binary relation describing elements of a set of *objects* by subsets of *attributes* (properties). Table 1 is an example of context where the objects are the solar system planets and the attributes are the properties of the planets. Each planet is described by its properties.

**Association Rules** AR is a data mining task with a well-documented rationale [1]. An *association rule* has the form:  $P \rightarrow C$ , where  $P$  and  $C$  are sets of attributes.  $P$  is called the *premise* of the rule and  $C$  the *conclusion*. In the context of Table 1, *far from the sun*  $\rightarrow$  *with moons* is an association rule, which means that to be far from the sun implies to have moons. An association rule is only an assertion; it may suffer exceptions. Statistical indicators measure the relevance of association rules (see Section 3).

**Formal Concept Analysis** In FCA [6], the set of all objects that share a set of attributes is called the *extent* of the set of attributes. The set of all attributes shared by all elements of a set of objects is called the *intent* of the set of objects. A *formal concept* is defined as a pair

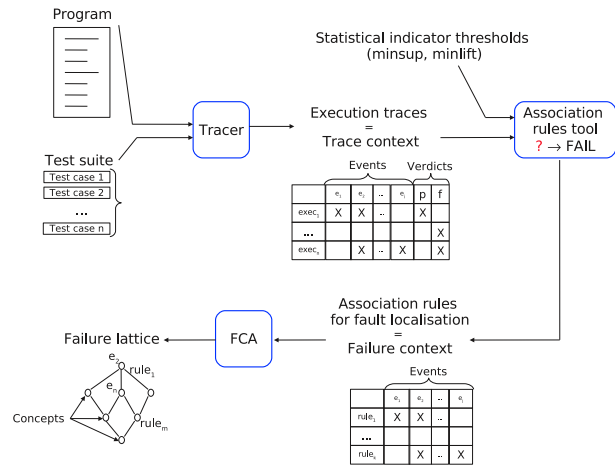


Figure 2. Process to build the failure lattice

(*set of objects, set of attributes*), where the set of objects is the extent of the set of attributes and the set of attributes is the intent of the set of objects.

The set of all concepts of a context can be represented by a concept lattice. The concepts are partially ordered in the *concept lattice*.  $X$  is a subconcept of  $Y$  if there is a path in the lattice between  $X$  and  $Y$  and  $X$  is below  $Y$ , it is denoted by  $X \leq Y$ . In the concept lattice, each attribute and each object labels only one concept. Namely, each object labels the most *specific* concept (i.e. with the smallest extent) to which it belongs. Each attribute labels the most *general* concept to which it belongs. Figure 1<sup>2</sup> shows the concept lattice associated to the context of Table 1. For example, the extent of concept A is {*Jupiter, Saturn, Uranus, Neptune*} and its intent is {*far from the sun, with moons*}. Regarding A's extent, Jupiter and Saturn label the more specific concept G and Uranus and Neptune label the more specific concept H. The attribute *with moons* labels the more general concept C. Thus only the attribute *far from sun* labels concept A.

## 3. Data Mining and Fault Localization

**A Lattice of Failures** The whole process is summarized in Figure 2. It aims at building what we call a *failure lattice*. The first step is the execution of the test cases. A summary of each execution is stored in a trace composed of *events*. In addition, we assume that each execution trace contains the verdict of the execution, *PASS* ( $p$ ) if the execution produces the expected results and *FAIL* ( $f$ ) otherwise. The multiset of all traces forms the *trace context*. The *objects* of the trace context are the test cases. The *attributes* are all

<sup>2</sup>This lattice and all the following ones are generated with the ToscanaJ tool (<http://toscanaj.sourceforge.net/>).

the events and the two verdicts,  $p$  and  $f$ . Each test case is described in the trace context by the events that belong to its trace and the verdict of the execution. Note that the extent of a set of trace events  $E$ ,  $extent(E)$ , denotes all the test cases that contain all elements of  $E$  in their trace. Thus, the extent of  $FAIL$ ,  $extent(\{f\})$ , denotes all the test cases that fail. The second step of the process aims at selecting events that appear often in failed executions and seldom in passed executions. From the trace context, specific association rules, called *fault localization rules*, are computed. The premise of a rule is a set of events and the conclusion is  $FAIL: E \rightarrow f$ .

We use two statistical indicators to select events that are involved in failed tests: support and lift. The *support* of a fault localization rule  $E \rightarrow f$  is the number of failed executions that have in their trace all events of  $E$ . It is defined as<sup>3</sup>:  $sup(E \rightarrow f) = \|extent(E) \cap extent(\{f\})\|$ . For the fault localization problem, a threshold of the support, *minsup*, indicates the minimum number of failed executions that should be covered by a rule in order to that rule be selected.

The *lift* of rule  $E \rightarrow f$  indicates if the occurrence of the premise increases the probability to observe a failed execution. It is defined as:

$$lift(E \rightarrow f) = \frac{sup(E \rightarrow f) \|All\ exec\|}{\|extent(E)\| \|extent(\{f\})\|}.$$

There are three possibilities. In the first case,  $lift(E \rightarrow f) < 1$ , executing  $E$  decreases the probability to fail (the premise *repels* the conclusion). In the second case,  $lift(E \rightarrow f) = 1$ , executing  $E$  does not impact the probability to fail ( $E$  and  $f$  are independant). In the last case,  $lift(E \rightarrow f) > 1$ , executing  $E$  increases the probability to fail (the premise *attracts* the conclusion).

The computed association rules are partially ordered by set inclusion of their premise. Let  $r_1 = E_1 \rightarrow f$  and  $r_2 = E_2 \rightarrow f$  be two rules; rule  $r_1$  is more specific than  $r_2$  if and only if  $E_1 \supset E_2$ . It is denoted by  $r_1 < r_2$ . In order to display the relations between the events of the program that have been filtered, the last step of the process builds the *failure context*. In that context, the objects are the association rules; the attributes are events. Each association rule is described by the events of its premise. The *failure lattice* is the concept lattice associated with the failure context. The failure lattice highlights the partial ordering of rules according to their premises.

**Statistical Indicators** The concepts in the failure lattice obey several interesting properties with respect to the statistical indicators. Firstly, the support of rules that label the concepts of the failure lattice decreases when exploring the lattice top-down. Thus, setting *minsup* equal to one object

<sup>3</sup>In the following,  $\|X\|$  denotes the cardinal of a set  $X$ .

```
[3393] int GetReal(double *reale,
      struct charac ** tp){
[3396] struct charac *curr,
      **curr_ptr = &curr;
[3397] int i = 0;
[3398] char num[MAX_REAL_LENGTH+1];
[3399] char ch;
[3403] *curr_ptr = *tp;
[3412] ch = TapeGet(curr_ptr);
[3416] if ((isdigit(ch) == 0)
      %% (ch != '+') %% (ch != '-'))
      %% (ch != ',')
[3417] return 13;
[3419] num[i] = ch;
[3425] i = i + 1;
[3426] ch = TapeGet(curr_ptr);

[3432] while (((isdigit(ch)
      || (ch == '.'))
      || (ch == 'e') || (ch == 'E')
      || (ch == '-'))
      %% ((*curr_ptr) != NULL)) {
[3433] if (i < MAX_REAL_LENGTH)
[3434] num[i] = ch;
[3438] i = i + 1;
[3439] ch = TapeGet(curr_ptr);
[3443] };
[3445] if (i >= MAX_REAL_LENGTH)
[3446] num[MAX_REAL_LENGTH] = '\0';
[3447] else
[3448] num[i] = '\0';
[3456] *reale = atof(num);
[3464] tp = curr_ptr; // FAULTY LINE
// Correct line: *tp = *curr_ptr
[3466] return 0;}
```

Figure 3. Excerpt of Mutant 6 of Space

is equivalent to searching for all rules that cover at least one failed execution. Setting *minsup* equal to the number of failed executions is equivalent to searching for a common explanation for all failures. We call *support cluster* a maximal set of connected concepts labelled by rules which have the same support value.

Secondly, the lift value strictly increases when exploring a support cluster top-down. The threshold of the lift, *minlift*, has a lower bound equal to 1, because when  $lift < 1$  a rule is not relevant. It also has an upper bound, *maxlift*, equal to  $\frac{\|All\ exec\|}{\|extent(f)\|}$ . In our approach, the lift indicates how the presence of a set of events in an execution trace improves the probability to have a failed execution. The lift threshold can be seen as a *resolution* cursor inside the support clusters. On the one hand, a low *minlift* implies a good resolution of the failure lattice, i.e. the events are separated, but it is costly because more rules are computed and thus more concepts. On the other hand, selecting a high *minlift* is cheaper in terms of number of computed rules and concepts but many events are aggregated.

**Example** We illustrate our approach on a faulty version of the Space program from the SIR repository. Space is written in C and contains 9126 lines of code (3638 of which are executable). For the example, events are the executed lines. An excerpt of the faulty code source is presented Figure 3. The fault, at line 3464, is related to a pointer. Instead of assigning the value of the memory pointed by  $tp$  to the memory pointed by  $curr\_ptr$ , both pointers point to the same memory space.

Figure 4 shows the related failure lattice for *minsup* = 86.79% and *minlift* = 1. Each concept of that lattice is actually labelled by a rule but we have omitted to display some of them for readability reasons. The failure lattice contains one support cluster only. The support value of this support cluster is 86.79% which corresponds to the proportion of failed executions for this case

As already mentioned, the rules are partially ordered in the failure lattice. For example, rule 1 is more specific than rule 2 (rule 1 < rule 2). Rule 1 contains in its premise all events that belong to the premise of

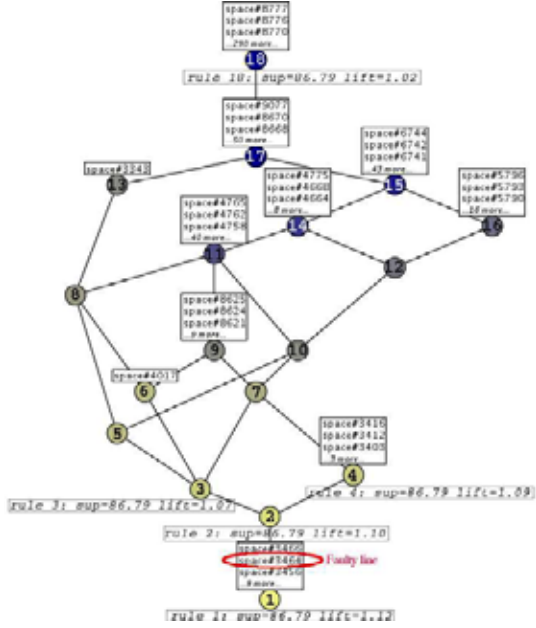


Figure 4. Failure lattice for Mutant 6 of Space

rule 2, i.e. all lines that label concepts above concept 2 (3416, 3412, 3403, 4017, ...). In addition, rule 1 contains in its premise 12 events that do not belong to the premise of rule 2: 3466, 3464, 3456, and nine others not shown. The most specific rules to explain the faults are at the bottom of the failure lattice. That’s why the failure lattice is explored bottom-up. The faulty line is in the most specific concept, at the bottom of the lattice. It is grouped together with eleven other lines. They are always executed together with line 3464 for this test suite. As can be seen on Figure 3, they all belong to the same basic block.

## 4. Experimental Study

We compare DeLLIS with existing methods on the Siemens suite. Then, we show that the method scales up for the Space program. DeLLIS uses a set of independent tools. The programs are traced with gcov<sup>4</sup>. The association rules are computed with the algorithm proposed in [3].

### 4.1. Siemens Suite Programs

In this section, we quantitatively compare DeLLIS to the methods for which results are available regarding the Siemens suite. These methods are Tarantula [8], Intersection Model (Inter Model), Union Model, Nearest Neighbor (NN) [11], Delta Debugging (DD) [4] and  $\chi$ Debug [12]. There are a total of 132 mutants of 7 programs (Table 2),

<sup>4</sup><http://gcc.gnu.org/online/docs/gcc/Gcov57.html>

Program	Description	Mutants	LOC	Tests
print_tokens	lexical analyzer	7	564	4130
print_tokens2	lexical analyzer	10	510	4115
replace	pattern replacement	32	563	5542
schedule	priority scheduler	9	412	2650
schedule2	priority scheduler	10	307	2710
tcas	altitude separation	41	173	1608
tot_info	information measure	23	406	1052

Table 2. Siemens suite programs

each containing a single fault on a single line. Let  $F_m$  denotes the fault of mutant  $m$ . Each program is accompanied by a test suite (a list of test cases). Some mutants do not fail for the test suites or fail with a segmentation fault. They are not considered by other methods, thus we do not consider them. There remains 121 usable mutants.

For the experiments, we set statistical indicator values such that the lattices for all the debugged programs are of similar size. We have chosen, arbitrarily, to obtain about 150 concepts in the failure lattices. That number makes the failure lattices easy to display and check by hand. Nevertheless, in the process of debugging a program, it is not essential to display rule lattices in their globality.

### Experimental Settings

We evaluate two strategies. The first strategy consists in starting from the bottom and traversing the lattice to go straightforwardly to the fault concept. This corresponds to the best case of our approach. This strategy assumes a competent debugging oracle, who knows at each step the best way to find the fault with clues. The second strategy consists in choosing a random path from the bottom in the lattice until a fault is located. This strategy assumes a debugging oracle who has little knowledge about the program, but is still able to recognize the fault when presented to him. Using a “Monte Carlo” approach and thanks to the law of large numbers, we compute an average estimation of the cost of this strategy.

**Metrics** We use the *Expense* metric of Jones *et al.* [7]:

$$Expense(F_m) = \frac{\|fault\_context(F_m)\|}{size\ of\ program} * 100$$

where  $fault\_context(F_m)$  is the set of lines explored before finding  $F_m$ . The *Expense* metric measures the percentage of lines that are explored to find the fault.

For both strategies, the best strategy and the random strategy, *Expense* is thus as follows:

$$Expense_B(F_m) = \frac{\|fault\_context_{Best}(F_m)\|}{size\ of\ program} * 100.$$

$$Expense_R(N, F_m) = \frac{1}{N} * \sum_{i=1}^N \frac{\|fault\_context_i(F_m)\| * 100}{size\ of\ program}.$$

$Expense_R$  is the arithmetic mean of the percentages of lines needed to find the fault during  $N$  random explorations of the failure lattice. A random exploration is a sequence of random paths in the rule lattice. A random path of the failure lattice is selected. If the fault is found on that path,

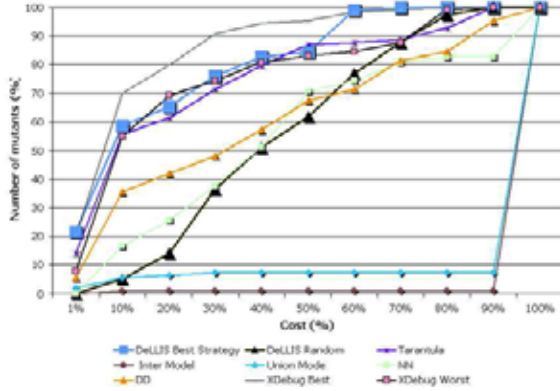


Figure 5. Frequency values of the methods

the execution stops and returns the fault context. Otherwise a new path is randomly selected, the previous fault context is added to the new fault context and so on until the fault is found. In the experiments, if after 20 selections the fault stays unfound, the returned fault context consists of all the lines of the lattice. We have noted that between 10 and 50, the computed results are not significantly different. Number  $N$  is chosen so that the confidence on  $Expense_R$  is about 1%. For any method  $M$ ,  $Expense_M$  allows to compute  $Freq_M(cost)$  which measures how many failures are explained by  $M$  for a given cost:  $Freq_M(cost) = \frac{|\{m | Expense_M(F_m) \leq cost\}|}{total\ number\ of\ mutants} * 100$ .

**Results**  $Freq_M(cost)$  can be plotted on a graph, so that the area under the curve indicates the global efficiency of method  $M$ . Figure 5 shows the curves for all the methods<sup>5</sup>. The DeLLIS strategies are represented by the two thick lines. For DeLLIS Best Strategy about 21% of mutant faults are found when inspecting less than 1% of the source code, and 100% when inspecting less than 70%. The best strategy of DeLLIS is as good as the best methods, TaranTula and  $\chi$ Debug, and the random strategy of DeLLIS is not worse than the other methods. We conjecture that the strategy of a human debugger is between both strategies. A very competent programmer with a lot of knowledge will choose relevant concepts to explore, and will therefore be close to the best strategy measured here. A regular programmer will still have some knowledge and will be in any case much better than the random traversal of the random strategy.

Whereas  $\chi$ Debug Best has better results, DeLLIS must be compared to  $\chi$ Debug Worst. Indeed, when the faulty line is grouped with other lines because they have the same score or they label the same concept, the faulty line is considered to be inspected *first* in  $\chi$ Debug Best, and *last* in  $\chi$ Debug Worst and DeLLIS. Therefore, our method is equivalent to

<sup>5</sup>The detailed results of the experiments can be found on: <http://www.irisa.fr/LIS/cellier/publis/these.pdf>

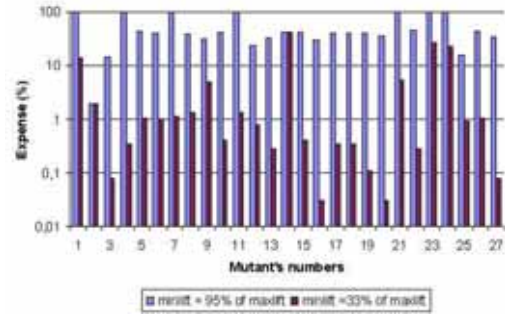


Figure 6. Expense values

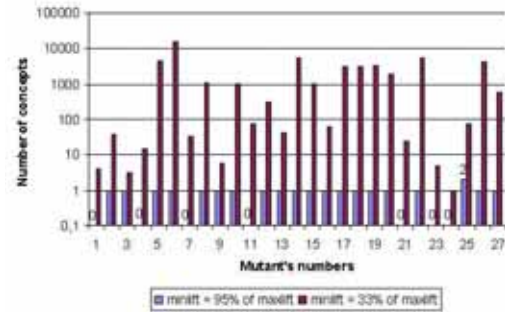


Figure 7. Number of concepts

the best methods dealing with fault on a single line, when comparing the number of lines to explore. Furthermore, DeLLIS gives more than a set of independent lines, it highlights the existing relations between them.

## 4.2. The Space Program

In this section, we study the behavior of our method on a program of several thousands lines, the Space program. In particular, we present how the expense value and the number of concepts of the best strategy vary with respect to the *minlift* value. Space has 38 associated mutants. SIR contains 27 usable mutants and 1000 test suites. For the experiment, we randomly choose one test suite such that, for each of the 27 mutants, at least one test case of the test suite fails.

In the experiments, the support threshold is set to the max value of the support. The mutants contain a single fault. The faulty line is thus executed by all failed executions. Different values of the lift threshold are set for each mutant in order to study the behavior of DeLLIS (8 values from 1 to  $(maxlift - 1) * 0.95 + 1$ ). We present in the paper two representative threshold values among the studied ones. The first lift threshold is a value close to the max value:  $(maxlift - 1) * 0.95 + 1$  (for *maxlift* see Section 3). The second lift threshold is  $(maxlift - 1)/3 + 1$ .

**Results** Figure 6 shows the expense values for each mutant when *minlift* is set to 95% of *maxlift* (light blue) and 33% of *maxlift* (dark red). The Expense value is presented in a logarithmic scale. The expenses are much higher with the larger *minlift*. When *minlift* = 95% of *maxlift*, some mutants, for example mutant 1, have an expense value equal to 100%, representing 3638 lines, namely the whole program. When *minlift* = 33% of *maxlift*, for all but 4 mutants, the percentage of investigated lines is below 10%. And for most of them, it has dropped below 1%. Note that 1 line corresponds to 0.03% of the program. Thus, 0.03% is the best Expense value that can be expected. Other experiments on intermediate values of *minlift* confirm that the lower *minlift*, the lower the expense value is, and the fewer lines have to be examined by a competent debugger.

When *minlift* = 33% of *maxlift*, DeLLIS, like Tarantula [8], is much better at detecting the fault than on the much smaller programs of the Siemens suite. For 51% of the versions, less than 1% of the code needs to be explored to find the fault. For 85% of the versions, less than 10% of the code needs to be explored to find the fault.

Figure 7 sheds some light on the results of Figure 6 and also explains why it is not always possible to start with a small *minlift*. The figure presents the number of concepts for each mutant when *minlift* is set to 95% of *maxlift* and 33% of *maxlift*. The number of concepts is also presented in a logarithmic scale. For *minlift* = 95% *maxlift*, for all but one mutant, either no rule or a single rule is computed. In the first case, the whole program has to be examined (Mutant 1). In the second case, the expense value is proportional to the number of events in the premise of the rule. For example, this represents 1571 lines for Mutant 5. When reducing *minlift*, the number of concepts increases and the labelling of the concepts is reduced. Traversing the failure lattice, at each step, fewer lines have to be examined, hence the better results for the Expense values with a low *minlift*.

However, for *minlift* = 33% of *maxlift*, for almost half of the mutants, the number of concepts is above a thousand and for one mutant it is even above 10000. Therefore, whereas Expense decreases when *minlift* increases, the cost of computing the failure lattices increases. Furthermore, when the number of concepts increases so does the number of possible paths in the lattice. For the best strategy this does not make a difference. However, in reality even a competent debugger is not guaranteed to always find the best path at once. Thus, a compromise must be found in practice between the number of concepts and the size of their labelling. At present, we start computing the rules with a relatively low *minlift*. If the lattice exceeds a given number of concepts, the computation is aborted and restarted with a higher value of *minlift* following a divide and conquer approach.

## 5 Conclusion

The DeLLIS process combines two data mining techniques, association rules and formal concept analysis in order to compute a failure lattice that gives clues for fault localization. The partial ordering of the lattice shows the dependencies between the code instructions. Experiments with the Siemens suite simulate a highly competent debugger and a mildly competent one. They show that, in both cases, DeLLIS quantitatively compares well with other methods. Experiments with the Space program show that the method scales up for programs of several thousand lines by tuning the lift threshold.

## References

- [1] R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *Int. Conf. Management of Data*, pages 207–216. ACM, 1993.
- [2] P. Cellier, M. Ducassé, S. Ferré, and O. Ridoux. Formal concept analysis enhances fault localization in software. In *Int. Conf. on Formal Concept Analysis*. Springer-Verlag, 2008. LNCS 4933.
- [3] P. Cellier, S. Ferré, O. Ridoux, and M. Ducassé. A parameterized algorithm to explore formal contexts with a taxonomy. *Int. J. of Foundations of Computer Science*, 2008.
- [4] H. Cleve and A. Zeller. Locating causes of program failures. In *Proc. of the Int. Conf. on Software Engineering*. ACM Press, 2005.
- [5] H. Do, S. G. Elbaum, and G. Rothermel. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering: An Int. J.*, 10(4):405–435, 2005.
- [6] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.
- [7] J. A. Jones, J. F. Bowring, and M. J. Harrold. Debugging in parallel. In *Proc. of the Int. Symp. on Software Testing and Analysis*, pages 16–26, July 2007.
- [8] J. A. Jones and M. J. Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *Int. Conf. on Automated Software Engineering*, pages 273–282. ACM, 2005.
- [9] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan. Scalable statistical bug isolation. In *Int. Conf. Programming Language Design and Implementation*. ACM Press, 2005.
- [10] C. Liu, L. Fei, X. Yan, J. Han, and S. P. Midkiff. Statistical debugging: A hypothesis testing-based approach. *IEEE Trans. Soft. Eng.*, 32(10):831–848, 2006.
- [11] M. Renieris and S. P. Reiss. Fault localization with nearest neighbor queries. In *Int. Conf. on Automated Software Engineering*. IEEE, 2003.
- [12] W. E. Wong, Y. Qi, L. Zhao, and K.-Y. Cai. Effective fault localization using code coverage. *Int. Computer Software and Applications Conf.*, 1:449–456, 2007.
- [13] W. E. Wong, Y. Shi, Y. Qi, and R. Golden. Using an rbfn neural network to locate program bugs. *Int. Symp. on Soft. Reliability Engineering*, 0:27–36, 2008.

# Extending AOP to Support Broad Runtime Monitoring Needs

Amjad Nusayr  
New Mexico State University  
anusayr@cs.nmsu.edu

Jonathan Cook  
New Mexico State University  
joncook@nmsu.edu

**Abstract**—Runtime monitoring, where some part of a program’s behavior and/or data is observed during execution, is a very useful technique that software developers use for understanding, analyzing, debugging, and improving their software. Aspect oriented programming is a natural fit for supporting the wide ranging instrumentation needs of runtime monitoring, but so far the limitations of AOP frameworks, namely supporting only code-based weaving and only a limited set of code feature joinpoint types, have hindered the application of AOP to many runtime monitoring needs. In this paper we present ideas for extending AOP to support weaving on dimensions other than source code, and for extending code-based weaving to finer-grained language constructs.

## I. INTRODUCTION

Aspect oriented programming is an elegant framework for constructing and implementing program behavior that is orthogonal to the underlying program code base. This type of *crosscutting concern* historically meant that the code implementing the behavior would be scattered around in the program and not localized nor modularized. From the beginning of AOP, it has been observed that runtime monitoring is a natural domain that benefits from AOP, with the “logging” monitor being the “Hello World” example for AOP. Instrumentation for runtime monitoring perfectly fits the idea of a crosscutting concern.

Despite this natural fit and several examples of such use ([4], [6], [9], [14], [20]), runtime monitoring has not seen a massive shift towards AOP systems. One could pessimistically say that this is because AOP is not “standard” in the industrial development toolkit; but even if the shipping product does not use AOP, AOP could still be used in the development environment to perform monitoring tasks needed during development.

Rather, we believe this stall is because of two reasons. One, existing popular AOP frameworks have not offered a great enough level of weaving *detail* to be more generally useful for runtime monitoring (e.g., statement coverage analysis needs more than just method call and field access weaving, and indeed the citation [4] above needed to extend the aspect language). Two, existing AOP ideas and frameworks have limited the axis of weaving to just the source code, excluding many monitoring needs that might not be easily translatable

to code-centric instrumentation (e.g., sampling-based profiling may need weaving based on execution time intervals rather than places in the code). We are not the first to note this; similar limitations were noted earlier in [20].

This paper presents ideas for extending the notions of AOP to include more detailed code-based weaving, and to include weaving over other dimensions of a program rather than just the source code. Others have noted and offered various unique additions and extensions to current AOP models (see Sections II and IV); our contribution here is a presentation of the scope and structure of extensions needed for runtime monitoring in particular.

## II. BACKGROUND

### A. Runtime Monitoring

Runtime monitoring is the act of observing an executing system in order to learn something about its dynamic behavior. Runtime monitoring generally refers only to the *act* of monitoring, which is in turn typically employed by some higher level analysis [2], [11], [21].

Runtime monitoring requires some sort of *instrumentation*, which probes the executing system and reports back something about the data or execution of that system. Instrumentation is often inserted *in-line* into a program, so that each time the program hits a certain point of execution, the instrumentation is executed. Sometimes, such as with debugger breakpoints, the instrumentation is hardware-assisted to make it much more efficient. Some monitoring approaches may use timer-based instrumentation, such as sampling-based code profiling (e.g., [16]), while others may use memory page protections to signal an event (e.g., [22]), and still others use probabilistic or counter-based sampling (e.g., [15]).

If a monitoring or higher-level analysis computation is not negligible, the instrumentation may simply record some minimal information about the event and store it for later *offline* processing, or deliver it through a queue to another thread or process that is performing the more intensive monitoring computation. Effectively creating the necessary instrumentation, and making it efficient, are complex and technologically difficult tasks, often involving the creation of specialized instrumentation tools by hand.

Monitors span the gamut from “barely noticeable” to “extremely painful” in terms of their impact on application performance. Sampling-based code profilers are extremely efficient, gathering very simple data periodically while the program

This work was supported by the National Science Foundation under grant CCF-0541075. Views expressed herein are those of the authors and do not necessarily represent those of the Foundation.

executes, and performing offline analysis to construct the execution profile [16]. Debugger expression watchpoints, on the other hand, can cause 4-6 orders of magnitude slowdown if they have no hardware support and thus need to evaluate an expression after each instruction [18]. Complex analyses such as the invariant inference of Daikon [8] require heavily instrumented programs to collect data variable history, and also use offline analyses to run the inference algorithms.

### B. Aspect Oriented Programming

Aspect Oriented Programming frameworks also *instrument* an underlying base program, but in AOP this purpose is more generic, to weave in any crosscutting functionality that should be factored out of the base program and not be replicated in the many locations in the program source where it is needed.

A basic AOP model defines some specific fundamental *pointcut designators* (PCD's),<sup>1</sup> which are features in the program execution where the advice of an aspect can be weaved in. A composition language allows a *pointcut expression* to combine and constrain these to define a *pointcut*, which is a set of program joinpoints that satisfy the expression, and where the advice will be woven in.

In existing AOP frameworks, the fundamental pointcut designators are chosen somewhat pragmatically: they must be actually useful to an aspect programmer, but they must also be relatively practical to implement in the AOP system. Thus, in existing AOP systems, pointcut designators are typically points in the program where inserting instrumentation is “not too hard”; for example, method calls are very often used as one of the fundamental pointcut designators.

The most popular AOP system, AspectJ, implements AOP for Java programs. Its pointcut designators include method calls, method executions, object field accesses, exceptions, and a few others [13].

### III. EXTENDING AOP IN DIMENSION AND DETAIL

There is a clear synergy between the ideas of AOP and the needs of runtime monitoring, and the only barrier to allowing AOP to be the unifying framework for almost all of runtime monitoring is the fact that current AOP frameworks are far too limited in the level of detail they support weaving at, and the fundamental style of weaving they support. The heart of our contribution here is to detail the extensions to current AOP ideas that will enable AOP to support the vast majority of monitoring needs. This paper is presenting initial ideas for these extensions, and thus the tone of this section is speculative; much more work is needed to experiment with and refine the specific forms and implementations of the various proposed extensions. In [17], a short workshop position paper, we very briefly formulated the idea of extending AOP into heretofore overlooked dimensions in order to support runtime monitoring. Here we elaborate and refine those initial ideas.

<sup>1</sup>Various literature refers to the basic built-in weaving-point designators of an aspect language as *joinpoint types*, *primitive pointcuts*, and *pointcut designators*. We use the latter term in this paper.

The general view of AOP thus far has been a one, or at most two, dimensioned view of aspects: they get woven into a base program based on joinpoints that are *code based* or *data based*. That is, the built-in set of pointcut designators that can be part of a pointcut specification are based on programming language features: method call, field access, etc. By far most pointcut designators get mapped to code features, and even the data-access pointcut designators often get mapped to code-based concepts (i.e., all expressions that access an object field).

We propose a more extensive *multi-dimensioned* view of AOP. Aspect weaving should not be limited to only be done based on code features, and this is especially important for AOP to support the broad needs of runtime monitoring. We identify four dimensions of weaving that are needed.

- **Code:** The traditional weaving over code features.
- **Data:** Weaving over concepts in data space.
- **Time:** Weaving based on time constraints.
- **Sampling:** Weaving that supports sampling-based instrumentation.

Each of these is detailed below. In describing the concepts, we are not concerned at all with whether or not they are difficult to implement. We only mean to offer the idea, and justify its potential utility.

1) *Code:* Code dimension weaving is well understood already, and most AOP frameworks support a variety of code-based pointcut designators. We only mention here that, for monitoring purposes, existing AOP frameworks are still quite limited in the features that they support; statement level coverage analysis and many other analyses depend on being able to instrument down to the statement (or basic block) level, yet current AOP systems do not support this. Analyses integrated with various testing methods may need to capture expression and sub-expression evaluation results, and other analyses need similar detail in what they can capture; for AOP to support such monitoring, the level of detail in terms of program concepts that is exposed to advice weaving must be much greater than current practice allows.

2) *Data:* Data dimension weaving has some, but very limited, support in existing AOP frameworks. For example, AspectJ has pointcut designators for object field accesses, but not for accesses to local variables, array elements, or arguments. Above the basic concepts of individual variables, the data dimension could support weaving on higher level data-oriented concepts, such as when a node in a data structure is accessed, when references to objects have changed, or how much space an application has allocated. In the allocation example, garbage collection algorithms that trigger on space usage could be seen as an aspect in the data dimension: when the application's space usage reaches the next triggering level, the garbage collector is triggered to execute. Thus conceptually, AOP over non-code dimensions can be a unifying theory for some current practices such as garbage collection and other services, which others have already noted [5].

3) *Time:* The time dimension will weave aspect code not based on location in code but on timers, either relative or absolute. An obvious example of the utility of this dimension



is profiling, where a timer-based interruption of the program samples where the program is at that point in time, and constructs a statistical profile of the execution behavior of the program. Other time-based uses would be to periodically check data structure health or application progress. This might be done over relative time, such as for every 10 minutes of program execution time, or it might be over absolute time, such as at 1:00am every Sunday night while the system is more likely to be idle.

Further refinement of this idea could support specific features of the particular programming languages and frameworks it is deployed on. For example, in Java distinct timers per thread might be useful, or per class or per object (an aspect might be “quiescent” until an object has had X amount of computational time on it, then is triggered and is allowed to perform a check on the object’s state or deep state).

4) *Sampling*: The sampling dimension controls whether or not the aspect is actually executed where woven, or not. Current AOP assumes that every time a joinpoint satisfying the pointcut expression is reached, the aspect will execute. However, research in runtime monitoring has shown the utility of *sampling* based approaches, where instrumentation is executed probabilistically, either randomly or (more efficiently) with a countdown/reset approach. AOP system could support such monitoring needs inherently, by allowing weaving to be done in this dimension.

Other types of counting are useful for certain runtime monitoring situations. For coverage analysis without regards to profiling, only a boolean “hit” flag is needed for each basic block. Once the flag is set, no more monitoring for that block is needed. Thus, being able to specify that an aspect execute on only the first instance (or the first N for other scenarios) would allow AOP to support this type of monitoring. Similarly, some applications need to be allowed to start up and get past an initialization or start up phase before it is useful to monitor them. In this case, skipping the first N executions of an aspect would be useful.

Many monitoring needs can benefit from “partial” monitoring, from single sites that just need to control the overhead of monitoring, to remote monitoring of user sites where small portions of each user site can be monitored efficiently, and results accumulated to understand the application as a whole. The sample dimension would support this.

5) *Integrating the dimensions*: Each of the four dimensions above have been shown by the examples given to be useful axes on which to extend AOP for runtime monitoring. We believe these dimensions and extensions will also be useful for other applications of AOP, but we only focus on runtime monitoring here. The question remains, though, on how they integrate and interact with each other. As with existing AOP pointcut designators, not all compositions of new PCD’s in these dimensions would make sense, and the semantics of some compositions need to be carefully specified.

The time dimension pointcut designators are usable by themselves; in this case, the aspect would be executed at a particular time, regardless of where the program is. This means

that the aspect could not assume some specific program state or scope, and thus the reflective information available to the aspect would be limited, or at the very least difficult to access. Composing the time pointcut designators with others, such as method invocations, brings in some design decisions because it is virtually guaranteed that the two pointcut designators, each designating an “instantaneous” event, would never be simultaneously satisfied. More work is needed to design useful semantics, but one approach would be to make the time domain designator mean “at least this much time” when composed with other designators.

Data dimension pointcut designators that track space usage are similar to existing ideas in triggering garbage collection, heap compaction, and other runtime efficiency services. Notice that pointcut designators that deal with space usage reaching a particular level must be *edge-triggered* rather than *level-triggered*. That is, the aspect executes when the condition first becomes true, and not continuously thereafter.

The sampling dimension is not usable by itself, but would depend on being composed with other pointcut designators that provide an underlying “real” joinpoint set over which to perform the count or the probability. It could naturally be composed with existing pointcut designators, and with both the time and data proposed pointcut designators, although some combinations may not be obviously useful.

#### IV. RELATED WORK

There is much recent activity and novel ideas for extending AOP in manners similar to our work here, but we do not find previous work that specifically laid out the ideas of various weaving dimensions and types of detailed code weaving that are particularly useful for runtime monitoring. Rajan [19] and then Dyer and Rajan [7] are investigating very similar ideas, explicitly working on arguing for more extensive join point models (thus allowing more pointcut designators) and embodying those in an intermediate language and virtual machine support for weaving. Rajan and Sullivan were, as far as we can tell, the first to make a clear note that current AOP models are insufficient to support many monitoring tasks such as coverage and profiling [20].

Harbulot and Gurd [10] used *abc* to extend AspectJ with a pointcut designator for loops, specifically focusing on numerical loops in scientific code. Bodden and Havelund [4] created a race detection tool, also finding that the existing set of pointcut designators was insufficient for their monitoring needs, and implemented their own new pointcut designators to specifically monitor locks and thus detect potential race conditions. Khaled et al. [12] used AspectJ for program monitoring, specifically for supporting program visualization. Hamlen and Jones [9] used AOP for the security monitoring of references, where security policies are checked in-line with the reference access. Bockisch et al. [3] describe VM support for dynamic join points, and this work may be able to provide the underlying capabilities needed for supporting time and data space dimension pointcut designators, and perhaps the probability/counting dimension as well.

A very nice formal framework for *Monitor-Oriented Programming* was detailed in [6]. This work describes the monitoring task in high-level formal notations, and demonstrates how AOP can be used to provide a rigorous framework for building runtime verification analyses. However, from the perspective of monitoring instrumentation, the MOP work in some sense *skipped* the hard part, by simply implementing their ideas on top of AspectJ. This means that all of the ideas are limited *in practice* to be usable only at the level of detail AspectJ provides: method call/return, field access, and a few other program events. Our contribution in this paper is relatively orthogonal to the MOP work; enabling MOP to use our extensions would produce a very powerful monitoring and verification framework.

## V. CONCLUSION

This paper presented ideas for extending the normal AOP concepts to support the full range of runtime monitoring needs. We showed that new pointcut designators that operate at a finer level detail over the base program's code are needed, and we proposed that AOP support dimensions of weaving other than the source code: data, time, and sampling. We believe that these and other new ideas for aspect weaving will serve to enable AOP to be used for a large number of program monitoring tasks. This will move runtime monitoring from being dependent on highly technical instrumentation requirements to being generally available to developers who need particular monitoring tasks. Although not addressed in this paper, we also imagine that these new dimensions of weaving, and added detail of code-dimension weaving, will be useful for other purposes that AOP supports. The ideas of different dimensions also open up new realms of thinking about dynamic weaving and runtime support, and other parts of "typical" AOP frameworks.

Many of our ideas are still preliminary and need not only implementation, but also experimentation and refinement. We are continuing to work on implementing various pointcut designators, in particular more of the code-based designators. We are also beginning to experiment with using the designators to support common runtime monitoring tasks, and using this experience to refine the designators and to figure out what types of information are needed in the advice used for monitoring. Another direction that needs investigation is improving the performance of monitoring advice, since controlling the overhead cost is generally a real concern in program monitoring. Ongoing work (e.g., [1]) that finds new ways to optimize advice execution may help make detailed runtime monitoring even more efficient and usable in the future. New mechanisms for making reflective information easier and faster to obtain in the advice code will also be needed, perhaps generating more static data during compilation so that advice can quickly access the data without costly runtime searches.

## REFERENCES

[1] P. Avgustinov, A. S. Christensen, L. Hendren, S. Kuzins, J. Lhoták, O. Lhoták, O. de Moor, D. Sereni, G. Sittampalam, and J. Tibble. Optimising AspectJ. In *PLDI '05: Proc. 2005 ACM SIGPLAN Conference*

on *Programming Language Design and Implementation*, pages 117–128, New York, NY, USA, 2005. ACM.

[2] T. Ball. The Concept of Dynamic Analysis. *SIGSOFT Softw. Eng. Notes*, 24(6):216–234, 1999.

[3] C. Bockisch, M. Haupt, M. Mezini, and K. Ostermann. Virtual Machine Support for Dynamic Join Points. In *AOSD '04: Proc. 3rd International Conference on Aspect-Oriented Software Development*, pages 83–92, New York, NY, USA, 2004. ACM.

[4] E. Bodden and K. Havelund. Racer: effective race detection using aspectj. In *ISSTA '08: Proc. of the 2008 international symposium on Software testing and analysis*, pages 155–166, New York, NY, USA, 2008. ACM.

[5] J. Bonér and E. Kuleshov. Clustering the Java Virtual Machine using Aspect-Oriented Programming. In *Proc. AOSD 2007 Industry Track*, 2007.

[6] F. Chen and G. Roşu. MOP: an Efficient and Generic Runtime Verification Framework. In *OOPSLA '07: Proc. 22nd ACM SIGPLAN Conference on Object Oriented Programming, Systems, and Applications*, pages 569–588, New York, NY, USA, 2007. ACM.

[7] R. Dyer and H. Rajan. Nu: a dynamic aspect-oriented intermediate language model and virtual machine for flexible runtime adaptation. In *AOSD '08: Proc. of the 7th international conference on Aspect-oriented software development*, pages 191–202. ACM, 2008.

[8] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao. The Daikon System for Dynamic Detection of Likely Invariants. *Sci. Comput. Program.*, 69(1-3):35–45, 2007.

[9] K. W. Hamlen and M. Jones. Aspect-oriented in-lined reference monitors. In *PLAS '08: Proc. of the third ACM SIGPLAN workshop on Programming languages and analysis for security*, pages 11–20, New York, NY, USA, 2008. ACM.

[10] B. Harbulot and J. R. Gurd. A Join Point for Loops in AspectJ. In *AOSD '06: Proc. 5th International Conference on Aspect-Oriented Software Development*, pages 63–74, New York, NY, USA, 2006. ACM.

[11] C. Jeffery, W. Zhou, K. Templer, and M. Brazell. A Lightweight Architecture for Program Execution Monitoring. In *PASTE '98: Proc. ACM Workshop on Program Analysis for Software Tools and Engineering*, pages 67–74, New York, NY, USA, 1998. ACM Press.

[12] R. Khaled, J. Noble, and R. Biddle. InsectJ: Program Monitoring for Visualisation using AspectJ. In *ACSC '03: Proc. 26th Australasian Computer Science Conference*, pages 359–368, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[13] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An Overview of AspectJ. In *ECOOP '01: Proc. of the 15th European Conference on Object-Oriented Programming*, pages 327–353, London, UK, 2001. Springer-Verlag.

[14] K. Kiviluoma, J. Koskinen, and T. Mikkonen. Run-time monitoring of architecturally significant behaviors using behavioral profiles and aspects. In *ISSTA '06: Proc. of the 2006 international symposium on Software testing and analysis*, pages 181–190. ACM, 2006.

[15] B. Liblit, A. Aiken, A. X. Zheng, and M. I. Jordan. Bug Isolation Via Remote Program Sampling. *SIGPLAN Not.*, 38(5):141–154, 2003.

[16] E. Metz, R. Lencevicius, and T. F. Gonzalez. Performance Data Collection Using a Hybrid Approach. *SIGSOFT Softw. Eng. Notes*, 30(5):126–135, 2005.

[17] A. Nusayr and J. Cook. AOP for the Domain of Runtime Monitoring: Breaking Out of the Code-Based Model. In *Proc. 2009 AOSD Workshop on Domain-Specific Aspect Languages*, page 4pp, 2009.

[18] M. Palankar and J. E. Cook. Merging Traces of Hardware-Assisted Data Breakpoints. In *WODA '05: Proc. 3rd International Workshop on Dynamic Analysis*, pages 1–7, New York, NY, USA, 2005. ACM.

[19] H. Rajan. A case for explicit join point models for aspect-oriented intermediate languages. In *VMIL '07: Proc. of the 1st workshop on Virtual machines and intermediate languages for emerging modularization mechanisms*, page 4, New York, NY, USA, 2007. ACM.

[20] H. Rajan and K. Sullivan. Aspect language features for concern coverage profiling. In *AOSD '05: Proc. of the 4th international conference on Aspect-oriented software development*, pages 181–191, New York, NY, USA, 2005. ACM.

[21] B. A. Schroeder. On-Line Monitoring: A Tutorial. *Computer*, 28(6):72–78, 1995.

[22] P. Zhou, F. Qin, W. Liu, Y. Zhou, and J. Torrellas. iWatcher: Efficient Architectural Support for Software Debugging. In *Proc. of the 31st International Symposium on Computer Architecture (ISCA)*, pages 224–235, 2004.

# Clustering of Defect Reports Using Graph Partitioning Algorithms

Vasile Rus<sup>1</sup>, Xiaofei Nan<sup>2</sup>, Sajjan Shiva<sup>3</sup>, Yixin Chen<sup>4</sup>

<sup>1,3</sup>Dept. of Computer Science, University of Memphis, Memphis, TN 38152

<sup>2,4</sup>Dept. of Computer and Information Science, University of Mississippi, University, MS 38677

<sup>1</sup>vrus@memphis.edu, <sup>2</sup>xnan@olemiss.edu, <sup>3</sup>sshiva@memphis.edu, <sup>4</sup>yichen@cs.olemiss.edu

## Abstract

*We present in this paper several solutions to the challenging task of clustering software defect reports. Clustering defect reports can be very useful for prioritizing the testing effort and to better understand the nature of software defects. Despite some challenges with the language used and semi-structured nature of defect reports, our experiments on data collected from the open source project Mozilla show extremely promising results for clustering software defect reports using natural language processing and graph partitioning techniques. We report results with three models for representing the textual information in the defect reports and three clustering algorithms: normalized cut, size regularized cut, and k-means. Our data collection method allowed us to quickly develop a proof-of-concept setup. Experiments showed that normalized cut achieved the best performance in terms of average cluster purity, accuracy, and normalized mutual information.*

## 1 Introduction

We address in this paper the challenging task of clustering defect reports. Defect reports are detailed descriptions in natural language of defects, i.e. problems in a software product. The proper handling of defect reports throughout the testing process for various purposes, such as fixing bugs in the case of developers, could have a great impact on the quality of the released software product. The defect reports are currently created and analyzed manually by testers, developers, and other stakeholders. Manual analysis is tedious, error-prone, and time consuming, leading to a less efficient testing process.

Defect reports are filed by testers (or users) who discover the defects through testing (or use). Reports include many details: an *id* that uniquely identifies the defect, the *status* of the defect (e.g. new, verified, resolved), a *summary* field, and a *description* field. The description field is the richest source of information about the defect. The field describes

details in plain natural language about the defect, including symptoms and steps to reproduce the defect. The summary field is a one-sentence description of the problem.

We propose here advanced methods for clustering defect reports that take advantage of the description and summary fields of the reports. We regard each defect report as a textual document and use a well-known technique in information retrieval (IR), called the *vectorial representation* [1], to represent documents. As clustering algorithms, we applied the following three algorithms: k-means [4, 18], normalized cut [15], and size regularized cut [2]. This work extends our previous work on clustering defect reports in which we only experimented with the *k*-means clustering algorithm [13].

Three models were used to represent defect reports, one based on the summary field alone, one based on the description field alone, and another based on the union of both. Our experimental data consists of defect reports collected from the open source Mozilla project ([www.mozilla.org](http://www.mozilla.org)). However, the proposed methods are transferable to defect reports from other projects, e.g. Eclipse ([www.eclipse.org](http://www.eclipse.org)). The clustering was evaluated based on reports describing the same underlying problem. That is, defect reports are in the same cluster if they describe the same underlying problem.

As a preview of our results, we found that the normalized cut clustering algorithm [15] proved to be by far the most successful. Furthermore, using the union of the summary field and description field for clustering is better than using either the description field alone or summary field alone.

## 2. Related Work

There are two major lines of previous research relevant to our work: research on defect clustering, and research on using natural language processing (NLP) and information retrieval (IR) to mine artifacts from software repositories.

Clustering is the unsupervised classification of data points (usually represented as vectors in a multidimensional space) into groups (clusters) based on similarity [19]. The clustering problem has been addressed in many contexts

and by researchers in many disciplines. While we are not aware of any particular work on clustering defect reports, there is published research related to clustering defects in the manufacturing of semiconductors [6] and integrated-circuits (IC; [16]). Karnowski et al. [6] showed that fuzzy logic can help better cluster defects on semiconductor wafer maps. Singh and Krishna [16] have shown that using clustering information in optimization testing can significantly improve the shipped product.

The usage of NLP applications to improve software development and testing has been around at least since 1990s [14, 10, 11, 3]. More recently, there has been renewed interest in applying natural language techniques to mine useful artifacts from the various repositories associated with software projects (see the yearly Workshop on Mining Software Repositories at <http://msr.uwaterloo.ca>).

We discuss next a series of research efforts that are directly related to our work on clustering defect reports. Linstead et al. [8] described a framework to automatically mine developer contributions and competencies from a given code base, and extract software functions in the form of topics. Weiss and his colleagues [17] used  $k$  nearest neighbor to search for similar historical bug reports and further predict the fixing efforts. In other related work, annotation graphs have been used to identify bug-introducing changes [7] and different classification approaches, including Bayesian model, support vector machine, classification trees, and  $k$ -nearest neighbor, were tried for classifying software maintenance requests by Lucca et al. [9]. The use of the vectorial representation [1] to address the task of duplicate defect report identification has been investigated by Runeson et al. [12]. In this paper, we use the vectorial representation for the clustering of software defect reports. The clustering uses spectral graph partitioning algorithms, which are described in Section 3.

### 3. Spectral Graph Clustering

In recent years, spectral clustering based on graph partitioning theories has emerged as one of the most effective data clustering tools. Normalized cut (Ncut; [15]) is a graph bipartition method that attempts to organize nodes into groups so that the within-group similarity is high and the between-group similarity is low. Another graph partitioning method is size regularized cut (SRcut; [2]) which enables users to incorporate prior knowledge of the size of clusters into the clustering process and also minimizes the similarity between two clusters and, at the same time, searching for a balanced partition. Unfortunately, normalized cut and size regularized cut are both NP-complete problems. For Ncut, Shi and Malik proposed an approximated solution by solving a generalized eigenvalue problem [15]. As for SRcut, Chen et al. [2] proposed a relaxed

version of the optimization that finds the largest eigenvalue of an associated matrix and uses it to bipartition the graph. These two methods can be recursively applied to get more than two clusters. In this work, we tested two heuristics for the Ncut clustering and one heuristic for SRcut clustering. With the first heuristic, used with both Ncut and SRcut, the subgraph with the maximum number of nodes is recursively partitioned (random selection is used for tie breaking). With the second heuristic, the subgraph with the minimal cut value is bipartitioned.

## 4. Experiments and Results

In this section, we address in detail the issues of data representation, similarity measure, and evaluation metrics. We also present performance results on clustering defect reports collected from Mozilla's Bugzilla, its the defect database.

### 4.1 Defect Reports Representation

A first issue we must address is the logical view of the defect reports (see [1] for more information on logical view). We chose a representation in which we retain all the words (after applying some preprocessing steps) but no positional information. Furthermore, we experimented with three models for representing reports: using words in the summary, description, and the union of both. The advantage of using only the summary would be its relative small size, usually less than 50 words, which leads to fast clustering.

Another important issue to address is the formalism used for the representation. We used the vector space model [1]. A key feature in the vector space model is the weighting scheme of the words. We used the TF-IDF scheme (TF - term frequency; IDF - inverted document frequency).

### 4.2 Evaluation Metrics

Purity, accuracy, and normalized mutual information are our evaluation metrics. The purity of a cluster is the ratio of the dominant class size in the cluster to the cluster size itself. A larger value means that the cluster is a "purer" subset of the dominant class. We assign the dominant class of the documents within a cluster as the label of that cluster. A document is correctly clustered if its cluster label is identical to its class label provided by ground truth. The percentage of correctly clustered documents among the corpus is the accuracy. Purity and accuracy tend to increase with the number of clusters. However, mutual information is a measure that avoids this drawback. If normalized, mutual information values near 1 indicate that the similar partitioning, while a value close to 0 implies significantly different partitions.

### 4.3 Clustering Experiments

In our experiments, we chose to cluster bugs based on the fact that they describe the same defect. We regard a bug and its duplicates as a cluster. The data used in our experiments comes from Mozilla's Bugzilla, where accurate duplicate information about defects, as entered by human experts (developers), is available.

To create our experimental data set, we started collecting 20 Bugs from the *Hot Bugs List* of Mozilla's Bugzilla which contains the most filed recent bugs. We chose the top 20 defect reports from the *Hot Bugs List* in terms of largest number of duplicates and retrieved about 50 duplicates for each. We automatically collected the *Description* and *Summary* data of these bugs and stored them locally in text files. The final data set contained 1003 data points in 50 clusters. Some defect reports out of the 1020 that we collected initially (20 original defects at 50 duplicates each) were dropped because the description field was empty or was simply redirecting the user to another bug, e.g. the field contained textual pointers such as *see bug #123*. As such, the size of the clusters varies from 46 reports to 51 reports, i.e. we have approximately balanced cluster sizes.

For each report, three vectorial representations were created based on the description field, summary field, and the union of the two fields. The vocabulary size/ dimensionality for the three representations are 4569, 991, and 5128, respectively.

We applied three clustering algorithms, Ncut, SRcut, and  $k$ -means, to data set. It should be noted that the data set is balanced. That is, each cluster contains approximately same number of instances ( 50) as explained above. For the Ncut algorithm, two heuristics were tested to iteratively divide the data set into 20 clusters. In the first heuristic, named largest first (LF), the largest subgraph was divided in each iteration. In the second heuristic, named best first (BF), the subgraph with the minimal Ncut value was divided in each step. The SRcut used the first heuristic to iteratively generate 20 clusters. The  $\alpha$  parameter of the SRcut algorithm was chosen to be 0.8.

The three evaluation metrics, average purity, accuracy, and normalized mutual information are reported in Table 1. Because the results of  $k$ -means depend on the initial choice of the seeds, we repeated 20 runs of  $k$ -means on the data set and reported the average and standard deviation of each metric. From Table 1, we can draw the following conclusions.

- Ncut with the largest first heuristic outperforms, on any evaluation metric given above, Ncut with the best first heuristic, SRcut and  $k$ -means algorithms on all three vectorial representations. The only exception is the average purity metric for the summaries data. This proves that the largest first heuristic Ncut method is suitable for

balanced scenarios.

- The purity metric is biased towards smaller clusters. This is demonstrated by the  $k$ -means results. On all three vectorial representations, the average purity of  $k$ -means clustering is comparable to or higher than Ncut clustering. However, the accuracy and normalized mutual information of  $k$ -means are significantly lower than those of Ncut on all three vectorial representations. This is because  $k$ -means tends to generate a large number of small clusters.
- Using the combination of descriptions and summaries for clustering is better than using either representation separately.
- SRcut performed poorly on this data set. This is mainly due to the fact that the SRcut algorithm is designed for graph bipartition. When applied iteratively, it is difficult to find a proper value of the  $\alpha$  parameter that works in all iterations.

## 5. Conclusions and Future Work

We addressed in this paper the challenging task of clustering defect reports based on their textual descriptions, summaries, and both descriptions and summaries. Our experiments on defect reports from Mozilla's Bugzilla and with three clustering algorithms showed that normalized cut using a TF-IDF vectorial representation based on a combination of descriptions and summaries of reports leads to better clustering than using the summary or the description of defects alone. Our work has been motivated by our belief that the rich information in software defect reports, which are generated during the testing phase in the form of textual reports, can be of great value. For instance, if open defect reports are clustered and a resulting cluster seems to be large compared to the others then the testing effort should focus, during the next testing cycles, on the defects in the large cluster. The large cluster may be an indication of an extremely faulty component or connected components which generate many related defects.

We plan to continue our investigation of clustering defect reports by using other representations of the defect reports, e.g. using only the *overview* section of the description field of a software report, and other text and knowledge processing techniques, e.g. exploiting knowledge about the particular software product being developed. As each defect report has a specific structure, we are also interested in exploring clustering techniques that take into account the structure instead of treating a report as a bag of words.

## ACKNOWLEDGEMENT

The work of Rus and Shiva was sponsored by The University of Memphis under a Systems Testing Excellence

**Table 1. Comparisons of clustering performance for Ncut, SRcut, and  $k$ -means algorithms with balanced class-size. Three vectorial representations (VR) are used: description field (VR1), summary field (VR2), and the union of description and summary fields (VR3).  $Ncut_{LF}$  and  $Ncut_{BF}$  denote Ncut algorithm with largest first heuristic and best first heuristic, respectively.**

Evaluation Metric		Average Purity	Accuracy	Normalized Mutual Information
VR1	$Ncut_{LF}$	<b>0.8509</b>	<b>0.8235</b>	<b>0.8225</b>
	$Ncut_{BF}$	0.8554	0.7468	0.7778
	$SRcut_{LF}$	0.6448	0.6471	0.6479
	$k$ -means	<b>0.8566 ± 0.0240</b>	0.7579 ± 0.0382	0.7710 ± 0.0247
VR2	$Ncut_{LF}$	0.7856	<b>0.7677</b>	<b>0.7525</b>
	$Ncut_{BF}$	<b>0.8207</b>	0.7069	0.7124
	$SRcut_{LF}$	0.6215	0.6092	0.5938
	$k$ -means	0.7934 ± 0.0279	0.6252 ± 0.0250	0.6368 ± 0.0227
VR3	$Ncut_{LF}$	<b>0.8864</b>	<b>0.8614</b>	<b>0.8540</b>
	$Ncut_{BF}$	0.8665	0.7587	0.7906
	$SRcut_{LF}$	0.6767	0.6800	0.6845
	$k$ -means	<b>0.8879 ± 0.0232</b>	0.7899 ± 0.0354	0.8187 ± 0.0201

Program (STEP) project. The work of Nan and Chen was supported by The University of Mississippi.

## References

- [1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
- [2] Y. Chen, Y. Zhang, and X. Ji. Size regularized cut for data clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 18, MIT Press, Cambridge, pages 211–218, 2006.
- [3] L. Eitzkorn, L. Bowen, and C. Davis. An approach to program understanding by natural language understanding. *Natural Language Engineering*, 5(1):1–18, 1999.
- [4] J. Hartigan and M. Wong. A  $k$ -means clustering algorithm. *Applied Statistics*, 28, 100–108, 1979.
- [5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- [6] T. P. Karnowski, S. S. Gleason, and K. W. Tobin. Fuzzy logic connectivity in semiconductor defect clustering. In *Proc. SPIE Machine Vision Applications in Industrial Inspection VI*, A. R. Rao; Ning Chang; Eds., 3306, pages 44–53, 1998.
- [7] S. Kim, T. Zimmermann, K. Pan, and E.J. Whitehead, Jr. Automatic identification of bug-introducing changes. In *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, pages 81–90, 2006.
- [8] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. Mining Eclipse developer contributions via author-topic models. In *International Workshop on Mining Software Repositories*, 2007.
- [9] G. A. Di Lucca, M. Di Penta, S. Gradara. An approach to classify software maintenance requests. In *Proceedings of the International Conference on Software Maintenance (ICSM'02)*, pages 93–102, 2002.
- [10] P. Lutsky. Documentation parser to extract software test conditions. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 294–296 1992.
- [11] P. Lutsky. Using a document parser to automate software testing. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 59–63, Phoenix, Arizona, 1994.
- [12] P. Runeson, M. Alexandersson, and O. Nyholm. Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th International Conference on Software Engineering*, pages 499–510, 2007.
- [13] V. Rus, S. Mohammed, and S. Shiva. Automatic Clustering of Defect Reports. *Proceedings of the 20th International Conference on Software and Knowledge Engineering*, pages 291–297, 2008.
- [14] J. Schlimmer. Learning meta knowledge for database checking. In *Proceedings of the National Conference of the American Association of Artificial Intelligence (AAAI'91)*, pages 335–340, 1991.
- [15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- [16] A. Singh and C. Krishna. On the effect of defect clustering on test transparency and IC test optimization. *IEEE Transactions on Computers*, 45(6):753–757, 1996.
- [17] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller. How long will it take to fix this bug?. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, pages 1–8, 2007.
- [18] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.
- [19] R. Xu and D. Wunsch. *Clustering*. John Wiley & Sons, 2008.

# Documenting Quality Attributes of Software Components<sup>1</sup>

Wenhui Zhu, Yanchun Sun\*, Gang Huang, Hong Mei

*Institute of Software, School of Electronics Engineering & Computer Science, Peking University,  
Key Laboratory of High Confidence Software Technologies, Ministry of Education,  
Beijing 100871, P.R.China*

*E-mail: {zhuwh04, sunyc, huanggang}@sei.pku.edu.cn, meih@pku.edu.cn*

## Abstract

Quality attributes become much more important in today's component-based systems. A well-formed and easy-to-understand documentation on these quality attributes is necessary in component-based development. In this paper, we propose a Quality Attribute Specification (QAS) that utilizes the Trace Function Method (TFM). There are four parts to this specification: Quality Attribute Variables, Access Programs, Quality Attribute Variable Functions, and the Auxiliary Functions. Each of them addresses precise and needed details for developers. To demonstrate our approach, a case study regarding the design description is included. From our research, the three primary benefits that we attribute to using this specification are as follows: Firstly, it specifies what to measure, estimate, or calculate when such a request is proposed. Secondly, it stipulates the physical limitations of each quality attribute. Thirdly, it facilitates the description of qualitative attributes precisely.

**Index Terms** - Documentation, Quality Attributes.

## I. INTRODUCTION

Large software projects usually require many developers to work together within a period of time. To avoid confusions between engineers during the development, a precise document is often necessary [12]. A precise document is not only used for team's communication or even a personal development guide but also to ensure the end product with good quality. As the modern software systems become more complex, more functions and requirements make the development more challenging. Due to the increasing scale of the system, the communication that involves more developers and the same level of quality that is asked for more functions both bring the development difficulties. This trend also highlights the need for well-defined and precise document (or design description) to ensure software quality.

Quality attributes, which assess the quality of a software system, mean the large group of properties; for example,

dependability, usability, safety and security [21]. To build systems that deliver their intended quality, it is essential to systematically take quality attributes into account at design time and not as an afterthought during implementation. Past research about the description of the quality attribute do not have the capability of describing quality attributes precisely in details so that the description can become a good aid to the system development. For example, when a description says "average throughput", unless this term is unambiguous, the developers can not have clear and sufficient information to understand the relationships among the programs that realize it. These drawbacks become more obvious when the description is for complex software development.

We believe that, for each quality attribute, it needs to satisfy certain conditions and its implementation detail should be included in the design description [12]. In this work, we adopt Parnas' Trace Function Method (TFM) [11] to propose an approach, "Quality Attribute Specification (QAS)". It facilitates the description of quality attributes in a observable granularity and also expresses quality attributes in interface specifications. In our method, the intention, the type and the name of the quality attribute all can be clearly expressed along with the definitions of the relationships between the quality attributes and other interface elements [10].

In the remainder of the paper, we will review the related work, explore our approach and demo how to use it through a case study.

## II. RELATED WORK

Many studies have been done on the design description of quality attributes. They can be divided into two categories. One is service-based and its emphasis is placed on describing the quality attributes both in terms of the client's requirements and service offerings. This category includes the Quality of Service Modeling Language (QML) that models quality characteristics of services and defines the quality of service (QoS) for both client needs and provider offers [1]. The other one is usage or environment specific. Four approaches belong to this kind. First one is the Quality of Service on IDL (QIDL). It is suggested for contract programming and CORBA IDL [3]. Second is the Quality Description Language (QDL). It primarily includes the description of contracts, system condition objects, and the adaptive behavior of objects [2]. Third is the UML profile for Schedulability, the Performance and Time Specification (SPT). It extends UML

\* Corresponding author

1. This effort is funded by the National Basic Research Program of China (973) under Grant No. 2005CB321805, and the Science Fund for Creative Research Groups of China under Grant No. 60821003, and the National Natural Science Foundation of China under Grant No. 60773151, and the National High-Tech Research and Development Program (863) of China under Grant No. 2007AA01Z127, 2008AA01Z139.

with basic timing and concurrency concepts to express requirements and properties for conducting schedulability and performance analysis [5, 6]. Last is the UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE). It provides the ability to model Non Functional Property (NFP) so the user is allowed to define the Non Functional Properties [19, 20]. This one can also be thought as an extension of the SPT method.

### III. THE APPROACH

Our Quality Attribute Specification (QAS) approach that specifies the quality attributes for components learns from Trace Function Method (TFM) [11].

#### A. Concept

Trace Function Method (TFM) is the generic method describes the interface specification based on early algebraic, axiomatic and trace-based approaches. When TFM is used to describe the interface of the component, it hides the internal data structure that should not be mentioned in a specification. In the TFM, the communication of the component is described by those global or shared variables that are considered as a part of its interface with external software or hardware. For example, when a global or shared variable  $x$  is in the component interface  $f(x)$ , we should put them in the TFM. Many successful applications and examples can be found in [11]. Therefore, when the quality attribute is considered, TFM can be used to define what should be described in interface specification and which quality attribute associated variables should be thought as the global variables.

According to TFM, since those global or shared variables are considered as a means for component communication, they contain the relationship to the programs they invoke. Since they are used to invoke programs that are part of the interface, the corresponding value changes of the variables occur. We define an “event” as the value changes of the variables. As a result, the invocation of the interface procedure function is also considered as an event. The name of the procedure being invoked is treated as the value of an input variable. The value of each output variable of a component is described as a function of the history of events in TFM, also called trace.

#### B. Overview of the Approach

Following the principle of TFM, QAS consists of four parts: (1) Quality Attribute Variable Table describes the name, type and range of the attributes that are interpreted from quality attributes (e.g., quality attribute performance can be response time or throughput) as variables and we call them quality attributes variables; (2) Access Program Table describes the impact of the quality attributes, which is caused by the function call outside the component; (3) Quality Attributes Variables Function contains several tables. Each one contains just one quality attribute and precisely describes how the access programs affect the quality attribute; and (4) Auxiliary Function helps to define the abbreviations used in the Quality Attributes Function Tables. More details will be

shown in the next section.

#### C. Four Elements in the Specification

##### 1. Quality Attributes Variables Table (QAVT)

In this table, we define the quality attributes as variables for the component because the input of the component that influences its behavior could be considered as a variable. Each quality attribute is described by its name, type and range. From past studies, most quality description language defines quality attributes from these three aspects or just the name, type and value of the quality attributes, yet the three aspects are insufficient to define the quality attributes.

##### 2. Access Program Table (APT)

This table shows what access programs the quality attributes will be affected by. It consists of program names, inputs, and the abbreviated event descriptor. Since an “event descriptor” in the table is a record of all the values of the variable before and after the event. The “abbreviated event descriptor” here only lists the values of variables that are changed or accessed during the event.

##### 3. Quality Attributes Variables Function (QAVF)

Quality Attributes Variables Function contains a set of output function and specifies the value of each variable as function of the trace of the components history. Trace here means a sequence of event descriptors that begins immediately after the creation of the specified object. Using traces, one can not only describe the history of a procedure or an object but also eliminate representations or models of the internal state information. For any deterministic component, the value of an output after an event can be described as the function of the trace that describes the sequence of the events that affect that component.

##### 4. Auxiliary Function (AF)

A set of auxiliary function definitions describe the abbreviation notation that is used in the output function definitions.

#### D. Benefits of Using the Specification

Describing quality attributes using TFM has several advantages. First, the Quality Attributes Variables Function Table precisely presents how the value of the quality attribute is affected by the history of the event. In this way, if people want to evaluate quality attributes by measuring the value changes which is affected by the event (access program), it could be easy to know what to measure (or estimate, calculate). Second, physical limitation, for example, the maximum memory allocation for the component is no more than 256 mb (Table 1), is explicitly written in the Quality Attributes Variables Table. Lastly, it provides a way to describe the non quantitative quality attributes precisely by defining the relation between the quality attribute value and the history of the event.

### IV. CASE STUDY

We use a shopping cart component as an example. This



component offers programs that can be accessed by others (three access programs) - “addItem”, “showList” and “verifyID”. As the names indicate, “addItem” means adding an item to the shopping cart, “showList” represents showing all the items in the shopping cart and “verifyID” means verifying the user’s ID when needed. The quality attributes which we express in the case are response time, memory, and security.

Quality attributes can be classified as quantitative and non-quantitative ones. It should be easy to define quantitative ones. For example, the response time would be the time value difference between certain access programs that are called beforehand and afterwards while we set “time” as a variable. Non quantitative quality attributes are not easy to define; however, we can use TFM to describe the behavior.

In our case, we consider three variables - “rTime”, “mA” and “Sec” (see Table 1). “rTime” represents the response time, “mA” stands for the memory allocation, and “Sec” represents the security. Each has a physical limit or the requirement constraint of the attribute. For example, mA’s range is between 0 and 256 mb. It represents maximum 256 mb of the system memory that has been allocated for the component to use. Besides, “Sec” here is defined as an enumerable variable that uses “low, medium and high” to show the level of security.

TABLE 1 QUALITY ATTRIBUTES VARIABLES

Variable Name	Type	Range
rTime	<double>	{0...1073741824}
mA	<int>	{0...256}
Sec	<enum>	{low, medium, high}

//i.e., the maximum value of time is 3600000000 microsecond (1 hour)

Table 2 should be read from left to right. Each line shows that the characteristics of the access program. Because the first two lines are similar, we use “verifyID” as an example. Its input is “string” data type. After “verifyID” is accessed, the values of those three quality attributes, “rTime”, “mA” and “Sec” are changed. However, for “showList”, it does not need an input variable. It only shows the content of Value which is “arraylist” type. To note that we do not determine how those values change in this table. They are described in other tables (Table 3, 4).

TABLE 2 ACCESS PROGRAM TABLE

Program Name	‘Value	‘in	Abbreviated Event Descriptor
verifyID		<string>	(PGM:verifyID, ‘in, rTime’, mA’, Sec’)
addItem		<int>	(PGM:addItem, ‘in, rTime’, mA’, Sec’)
showList	<arrayList>		(PGM:showList, ‘Value)

To show the differences of the quantitative and non-quantitative variables in the QAVF table and their related AF, we use quantitative variable, “rTime”, and non-quantitative variable, “Sec”, for explanation.

In Table 3, the value of the “rTime” equals to 0 when no access programs accesses the component, which means, no time will be spent on the component. The second line needs to be read along with the AF since the associated expression showed in the QAVF table is defined in the AF. In this second line, when there are access programs accessing the component from outside, we need to check whether current event is a “noeffect” expression. If we check the precise meaning of

“noeffect” expression in the AF, one kind of the “noeffect” can be found. It is the input of the access program “addItem” does not exist in the item list range; that is, PGM (e) = addItem  $\wedge$  (‘in(r (e))). If the situation is “noeffect”, the “rTime” variable value does not change and still equals to the value before the “addItem” program that is called. The third line explains the situation that does not belong to “noeffect”. In this situation, the value of “rTime” changes when the access program “verifyID” calls from the outside of the component. This changed value (of the “rTime”) is equal to the value before the calling by the program “verifyID”. The last line means similar but only the program “verifyID” is replaced by the program “addItem”.

TABLE 3 QUALITY ATTRIBUTE “rTime” FUNCTIONS

rTime(T) =		
T = _		0
(T ≠ _) $\wedge$	noeffect(r(T))	rTime(r(T))
	$\neg$ noeffect(r(T)) $\wedge$	rTime(r(T))+verifyID.rTime
	PGM(r (T))=verifyID $\wedge$	rTime(r(T))+addItem.rTime
	PGM(r (T))=addItem $\wedge$	

T: Trace, history of the events  
r(T): the most recent event in the trace

invalid (e) = ‘in(e) = {, /, :, ?, *, <, >}
nonexist(e) = ‘in(e) != in.range
noeffect(e) =
( PGM(e)=verifyID $\wedge$ (‘in(r(e)) = invalid(e) $\vee$
PGM(e)=addItem $\wedge$ (‘in(r(e)) = nonexist(e) $\vee$
PGM(e) = showList $\wedge$ ‘Value(e) = null

FIG. 1 The Content of Auxiliary Functions

Table 4 describes the relationship between non-quantitative variable “Sec” and the access programs. The interpretation is very similar to the quantitative one that we demonstrate in Table 3. There’s slight difference from the third to fifth line. The third line indicates that setting the value of the “Sec” variable should be set to “Medium” if the access program is “verifyID”. The fourth and fifth lines show that if access program “verifyID” is not accessed before access program (i.e., in fourth line, it is “addItem” and in fifth line, it is “showList”.) is called, the “Sec” value equals to “Low”.

TABLE 4 QUALITY ATTRIBUTE “Sec” FUNCTIONS

sec(T) =		
T = _		Null
(T ≠ _) $\wedge$	noeffect(r(T))	‘Sec(r(T))
	PGM(r (T))=verifyID $\wedge$	Medium
	$\neg$ noeffect(r(T)) $\wedge$	Low
	PGM(r (T))=addItem $\wedge$	
	$\neg$ ex(PGM(p(T))=verifyID)	Low
	PGM(r (T))=showList $\wedge$	
	$\neg$ ex(PGM(p(T))=verifyID)	

r(T): most recent event in the trace  
p(T): the past events in the trace  
ex(e): the event exists

The example above explicitly shows that engineers can work out the precise quality attribute information with little difficulty by using the TFM method. A number of outcomes are found in the specification. Firstly, if engineers want to know the limitation of certain quality attribute, they can just check the “Range” section in the Quality Attribute Variable Table. The quality attribute range is clearly identified here. Secondly, when engineers are interested in which variables affect certain quality attribute, they can check the Access

Program Table. For example, when engineers only consider the value of the variable “Sec”, on the “Abbreviated Event Descriptors” column in the Table 2, they use “Sec” as the keyword to search and the result will be the information they require. Therefore, the search result of the keyword “Sec” contains the programs “verifyID”, “addItem”, and “showList”. They all influence the variable “Sec”. Finally, to find out how these access programs affect the quality attributes, engineers just have to look at Output Functions Table. If the attribute (variable) is quantitative (e.g. rTime) the amount of the value changes due to the call of the access program would become the target information; thus, engineers can understand what to measure (or estimate, calculate) if they want to know the value for this attribute. However, if the variable is non-quantitative like “Sec”, by reading the table, engineers should be able to get the information about a history of access program calls. The history should explain if the current “Sec” value is “Medium” or “Low”.

#### V. CONCLUSION AND FUTURE WORK

The research of the software quality plays a key role in modern software development due to the need of the stable running system with the complex functionalities. The attributes of the system, which are used to describe the related quality in the design document, are quality attributes. Previous methods that describe the quality attributes cannot provide enough information for the developer to guide their development of the system. Instead, our Quality Attribute Specification precisely defines the quality associated relationship between interfaces and their accessing programs; therefore, giving sufficient information to help developers to implement the system with the required quality. Our approach that learns from the TFM provides four tables for the developer to define their details of the quality attribute. From QAS, people can evaluate the quality attribute by investigating the value changes of it after the influence that is caused by the associated access programs. In this way, it could be easy to know what to measure (or estimate, calculate) and how to do it. Besides, the physical limitation of each quality attribute is explicitly expressed in the Quality Attributes Variables Table. Furthermore, those non-quantitative quality attributes can be described precisely by defining the relation between the trace of the events and the quality attribute value. Those benefits could be easily understood from our case study above.

Our work currently only focuses on addressing quality attributes of the component interface. However, when the components are composed into a system, all the quality attributes have to be considered as a whole; as a result, the complex system behaviors may bring a complicated effect on the quality attributes and is not very clear now. In the future, we would like to shift our focus to the architectural level, so we will try to describe the quality attributes on the software architecture. Therefore, how to express each quality attribute separately in software architecture precisely needs our future investigation.

#### VI. ACKNOWLEDGEMENT

This work receives invaluable inspiration and supports from Dr. David L. Parnas. We would like to appreciate for his help and guidance to our research.

#### REFERENCES

- [1] S. Frolund and J. Koistinen, “Quality of Service Specification in Distributed Object Systems”, *Distributed Systems Engineering Journal*, Vol. 5(4), (December 1998).
- [2] J. P. Loyall, R. E. Schantz, J. A. Zinky, and D. E. Bakken. “Specifying and measuring quality of service in distributed object systems”. In *1st International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 43 – 52. IEEE Press, April 1998.
- [3] C. Becker and K. Geihs. “Generic QoS-support for CORBA”. In *Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, page 60. IEEE Press, July 2000.
- [4] M. Born, A. Halteren and O. Kath, “Modeling and Runtime Support for Quality of Service in Distributed Component Platforms”, *Proc. 11th Annual IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, December 2000.
- [5] OMG. “UML Profile for Schedulability, performance, and Time Specification”. version 1.1, formal/05-01-02
- [6] OMG. “UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification”. Version 1.1, formal/2008-04-05
- [7] OMG. “A Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems”, Beta2. 2008-06-08
- [8] J. Aagedal and E. Ecklund, “Modelling QoS: Toward a UML Profile”, *Proc. UML-2002 Conference*, Springer Verlag (2002).
- [9] Selic, B., “A Generic Framework for Modeling Resources with UML,” *IEEE Computer*, Vol. 33(6), June 2000.
- [10] Wenhui. Zhu and David L. Parnas. “Using Documentation to Build a Self-Adaptive System”. In *International Workshop on Self-Adaptive Software Engineering at IEEE CSMR 08*.
- [11] David L. Parnas, Sergiy A. Vilkomir, “Precise Documentation of Critical Software”, *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE’07)*, Volume 00, September 2007, Page 237-244.
- [12] David L. Parnas “Document Driven Disciplined Development of Software”, *Proceedings of the 2005 Australian Software Engineering Conference*, 2005
- [13] J. Loyall, R. Schantz, J. Zinky and D. Bakken, “Specifying and Measuring Quality of Service in Distributed Object Systems”, *Proc. 5th International Symposium on Object-Oriented Real-Time Distributed Computing*, April, 1998.
- [14] C. Becker and K. Geihs “MAQS - Management for Adaptive QoS-enabled Services”, *IEEE Workshop on Middleware for Distributed Real-Time Systems and Services*, December, 1997.
- [15] Praveen K. Sharma, Joseph P. Loyall, George T. Heineman, Richard E. Schantz, Richard Shapiro, and Gary Duzan, “Component-Based Dynamic QoS Adaptations in Distributed Real-Time and Embedded Systems”, *CoopIS/DOA/ODBASE 2004*, LNCS 3291, pp. 1208–1224, 2004.
- [16] Miguel A. de Miguel, “QoS Modeling Language for High Quality Systems”, *Proceedings of The Eighth IEEE International Workshop on Object-Oriented Real-Time*, 2003
- [17] ISO. “Software engineering—product quality”, *ISO/IEC 9126:2001*
- [18] Simona Bernardi, Dorina C. Petriu. “Comparing two UML Profiles for Non-functional Requirement Annotations: the SPT and QoS Profiles”. *UML’2004*, SVERTS.2004
- [19] Huascar Espinoza, etc. “Annotating UML Models with Non-functional Properties for Quantitative Analysis”. *MoDELS’2005 workshop*. 2005
- [20] Gruiua-Catalin Roman. “Taxonomy of Current Issues in Requirement Engineering”. *IEEE 1985*
- [21] F. Manola, “Providing Systemic Properties (Ilities) and Quality of Service in Component-Based Systems,” *Object Services and Consulting, Inc.*, Technical Report 1999.

# Taming Inconsistency in Value-Based Software Development

Du Zhang

*Department of Computer Science  
California State University  
Sacramento, CA 95819-6021  
zhangd@ecs.csus.edu*

## Abstract

*The main objective in value-based software engineering is to align value considerations with the full range of existing and emerging software engineering principles and practices so as to increase value for software assets. However, inconsistencies that arise during software development pose a threat to the objective, if they are not properly tamed. In this paper, we review the current landscape of inconsistency management in the value-neutral software engineering setting. After examining the dimensions of inconsistency as a phenomenon in software development process, we propose some guidelines on how to tame conflicting descriptions and inconsistencies in value-based software engineering. The take-home message is that inconsistency is an issue to be reckoned with in value-based software engineering.*

**Keywords:** *value-based software engineering, stakeholder value propositions, inconsistency management.*

## 1. Introduction

Value-based software engineering (VBSE) is to integrate value considerations into the full range of existing and emerging software engineering principles and practices so as to increase the return on investment for the stakeholders and optimize other relevant value objectives of software projects [1, 2]. The VBSE paradigm calls for augmenting the technical activities with economic, management, and cognitive considerations when developing successful software systems. Therefore, the essence in VBSE is to align software development and maintenance with customer requirements and strategic business objectives. It offers a framework where the stakeholders value propositions (SVPs) are incorporated into the technical and managerial decisions made during software development and maintenance [1, 10].

A key concept in VBSE is valuation that ascertains the economic value of a product, a service, or a process [8]. Value can be: profits (generated from products), strategic positioning in market share, utility, relative worth,

reputation, customer loyalty, innovation technology, cost reduction, quality of life, or improved productivity [18]. Value can be created along economical, physical, emotional, social, cognitive and political dimensions [18]. The goal of the road map in VBSE is to make software development and maintenance decisions that are better for value creation [2].

Software engineering research and practice thus far are mainly conducted in a value-neutral setting where each artifact is treated equally important during a software development process. To help bring the economic value into the software development process, an emerging agenda of issues in VBSE has been proposed in [2], that includes the following areas:

- Value-based requirements engineering. The key objectives include recognition of success-critical stakeholders, elicitation of SVP, and reconciliation of SVPs.
- Value-based architecting. The goals are to iron out the discrepancy between a system's objectives and achievable architectural solutions.
- Value-based design and development. The goals are to ensure that a software system's objectives and its value considerations are embodied in the software's design and development practices.
- Value-based verification and validation. The objectives are to determine that a software solution meets its value objectives and that V&V tasks are sequenced and prioritized as investing activities.
- Value-based planning and control. The objectives in this area are to incorporate the value delivered to stakeholders into the product planning and control techniques.
- Value-based risk management. How to factor the value considerations into principles and practices for risk identification, analysis, prioritization, and mitigation is the main focus in this area.
- Value-based quality management. The goals are to prioritize desired software quality considerations with respect to SVPs.
- Value-based people management. The tasks involve building stakeholder team, manage expectations, and reconcile SVPs.

As a result of the essential difficulties (complexity, conformity, changeability, and invisibility) inherent in developing large software systems that were eloquently spelled out in Brooks' classic paper of "No Silver Bullet: Essence and Accident in Software Engineering" [6], inconsistencies or conflicting descriptions are bound to arise either among different segments of a specification, or among specifications at various levels of abstraction during software development. These inconsistencies, if not properly tamed, would result in value reduction or destruction for the software asset, thus threatening the main objective of value creation in VBSE. How to leverage or manage inconsistency in value-neutral software development drew significant attention and research since the nineties [7, 11-17, 19]. It is no exception that VBSE has to deal with the inconsistency issue as well. The purpose of this paper is to draw attention on the issue of how to tame conflicting descriptions and inconsistencies in VBSE.

The rest of the paper is organized as follows. Section 2 offers an overview of inconsistency management in value-neutral software engineering. Section 3 examines the dimensions of inconsistency. In Section 4, we describe some guidelines on how to augment results obtained in the value-neutral setting to tame conflicting descriptions and inconsistencies in VBSE. Finally Section 5 concludes the paper with remark on future work.

## 2. Related Work

Since the late eighties and early nineties, there has been an increased research on the issue of managing inconsistency in software development process. A number of journal special issues and conference sessions devoted their attention to the issue. As a result, various approaches and frameworks have been proposed. Here we only highlight some of the reported results.

A framework was described in [13] for managing inconsistency in software development. The cornerstone of the proposed framework is a set of consistency rules that describe some proper relationships that must be observed between or among a set of descriptions (analysis models, process models, specifications, design patterns, test plans, and so forth). A violation of a consistency rule by a set of descriptions is referred to as an inconsistency. There are two major components in the framework: inconsistency diagnosis, and inconsistency handling. The diagnosis component includes steps to locate, identify and classify an inconsistency. Measurement of inconsistency needs to be established to support the tasks. The handling component offers a number of strategies to resolve, tolerate, or ignore classified inconsistencies. To tolerate an inconsistency, there are options to defer, circumvent or ameliorate it. The handling component has to be augmented with a feedback loop on analyzing the impact

and risk of unresolved inconsistency during the development process.

The work in [17] reported results in inconsistency management in requirement engineering. The approach attempts to deal with inconsistency at the process (elaboration), product (requirements) and instance (running system) levels. A classification of inconsistencies in requirement engineering has been proposed that includes: process-level deviation, instance-level deviation, terminology clash, designation clash, structure clash, conflict, divergence, competition, and obstruction. Formal techniques and heuristics were proposed to resolve conflicts and divergences.

The results in [16] introduced the concept of overlapped models as the necessary condition for inconsistency. A framework was proposed that consisted of the following activities: detection of overlaps, detection of inconsistencies, diagnosis of inconsistencies, handling of inconsistencies, tracking, and specification and application of an inconsistency management policy. When handling inconsistencies, a course of action is committed only after possible actions are identified, the costs and benefits of each of such actions evaluated, and risks of non-action assessed.

The focus in [12] was to manage inconsistent specifications. An inconsistency handling approach was proposed that relied on a particular paraconsistent logic formalism and allowed continued specification development in the presence of inconsistency.

The results in [19] address the consistency issue in UML models. A set of consistency constraints is defined that establishes conditions the violations of which constitute inconsistencies in the UML models. The consistency constraints include the following types: intra-diagram (defined on a specific type of diagram of a model), inter-diagram (defined on two or more diagrams of the same type), inter-model (defined on diagrams of more than one type), horizontal (defined between diagrams of the same type at the same abstraction level), and vertical (defined between diagrams that have refinement relationship).

## 3. Dimensions of Inconsistency

As eloquently put by Brooks in [6], "The hardest part of the software task is arriving at a complete and consistent specification, and much of the essence of building a program is in fact the debugging of the specification." Inconsistency is an integral part of software development process and is of multi-dimensional characteristics. Conflicting descriptions can arise in different circumstances that are due to different causes, having different interpretations, manifesting themselves in different forms or types, of varying significance, detected through different methods, having different levels of desirability, and demanding different actions. There are a

number of deeper issues we need to contend with before making the appropriate decisions with inconsistency in software development process. Figure 1 summarizes the dimensions of inconsistency in VBSE.

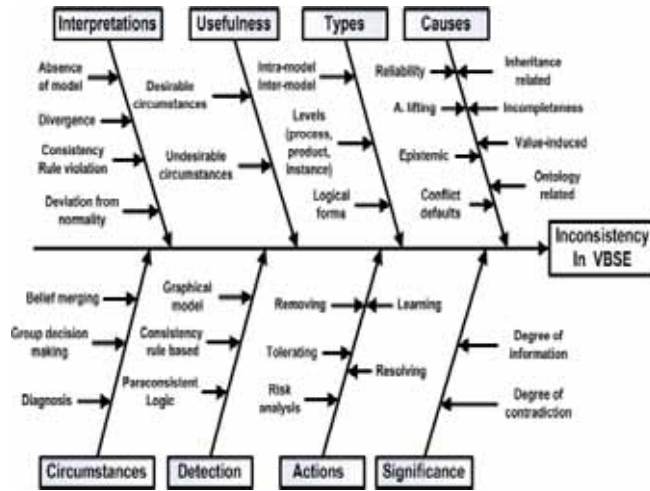


Figure 1. Dimensions of inconsistency in VBSE.

Due to space limitation, we will not be able to elaborate on issues in each of the dimensions. Instead, we will focus our attention only on the causes and a classification of inconsistency based on logical forms.

### 3.1. Causes

Inconsistency can arise in software development process in a number of different ways.

**Ontology related reasons.** These include circumstances such as: either the lack of explicit or implicit constraints in the ontology specification (e.g., an ontology does not specify that animals and vegetables are mutually exclusive and jointly exhaustive in living things), or the discrepancy in terminology and its usage in different agents (e.g., polysemy, or antonymy).

**Epistemic conflicts.** These types of antagonism stem from the fact that different agents, developers, or stakeholders have their beliefs that are incompatible with each other.

**Conflicting defaults.** In order to reason about the application domain a software system is to be deployed in, some general assumptions need to be made through *defaults* and *exceptions*. Default reasoning is an important characteristic of software development process, since defaults are temporary assumptions that can later be revised as new information becomes available. Defaults can become conflicting with each other.

**Lack of complete information.** In default reasoning, a developer may need to augment the set of base beliefs with the negation of any ground atom that is not entailed by the base set. This is what is referred to as the *closed-world assumption* (CWA). Thus CWA allows for an augmented theory about a domain that consists of the

original set of base beliefs plus a set of negative ground literals.

An agent's knowledge is *complete* if and only if for every sentence  $\alpha$  in its vocabulary, either  $\alpha$  or  $\neg\alpha$  is known [5]. In general, an agent's knowledge can be incomplete. For instance, if an agent's knowledge contains the following:  $P(a) \vee Q(a)$ , then the agent cannot deduce either  $P(a)$  or  $\neg P(a)$  (neither can it deduce  $Q(a)$  or  $\neg Q(a)$ ). Under CWA, this would result in inconsistency in the agent's knowledge. For example, given the sentences in an agent's knowledge:  $\{P(x) \vee Q(x), P(a), Q(b)\}$ , if there are three constants  $\{a, b, c\}$  in the domain, then there is inconsistency under CWA in the knowledge because the augmented theory contains the following:

$$\{P(c) \vee Q(c), \neg P(c), \neg Q(c)\}$$

**Defeasible inheritance induced.** Property inheritance is an important mechanism in dealing with abstracting, classifying and generalizing knowledge that is organized in a hierarchical structure. Property inheritance is regarded as a form of default reasoning [5]. Inheritance where inherited properties can be overridden is referred to as *defeasible inheritance* [5]. Inconsistency may arise in defeasible inheritance.

**Conflict of value propositions.** This type of inconsistency stems from contradictory value propositions of stakeholders. SVPs tend to be emergent and are not necessarily compatible with each other.

**Assertion lifting.** If an agent's knowledge is compartmentalized into *contexts*, lifting or importing assertions from one context to another may result in knowledge in a particular context becoming inconsistent.

**Concept forming process in description logics.** Inconsistency can arise from the concept forming process in *description logics*. There are concept-forming operators in description logics such as: ALL, EXISTS, AT\_MOST, FILLS, AND. Their use in constructing compound concepts may result in conflicting conditions. For instance, the following concept through the AND operator is inconsistent.

$$[\text{AND} [\text{EXISTS } 3 \text{ agent}] [\text{AT\_MOST } 2 \text{ agent}]]$$

If a concept  $d$  is inconsistent, then conjoining the two  $[\text{ALL } r \ d]$  and  $[\text{EXISTS } 1 \ r]$  results in inconsistency [5].

**Reliability of information source.** When the source of information is not reliable, or the information is not up-to-date, inconsistency can arise.

**Deliberate act.** Inconsistency can also be the result of some deliberate act as an integral part of an overall strategy in accomplishing a particular objective.

### 3.2. A Classification

Definitions are provided below for a list of possible cases of inconsistency that can arise in software development process, though the list is not meant to be complete. The types and notations of those cases of inconsistency are summarized in Table 1.

Table 1. Types of inconsistency.

Inconsistency Type	Notation
<i>Complementary</i>	$L_1 \neq L_2$
<i>Mutually exclusive</i>	$L_1 \neq L_2$
<i>Incompatible</i>	$L_1 \not\equiv L_2$
<i>Anti-subtype</i>	$L_1 \not\sqsubseteq L_2$
<i>Anti-supertype</i>	$L \not\supseteq (\sqcup L_i)$
<i>Asymmetric</i>	$L_1 \not\psi L_2$
<i>Anti-inverse</i>	$L_1 \neq L_2$
<i>Mismatching</i>	$L \not\equiv (\sqcap L_i)$
<i>Disagreeing</i>	$L_1 \not\cong L_2$
<i>Contradictory</i>	$L_1 \neq L_2$
<i>iProbVal</i>	$\text{Prob}(L) \not\subseteq \text{Prob}(\Delta)$

**Definition 1.** Given  $A_1$  and  $A_2$  as syntactically identical atoms (same predicate symbol, same arity, and same terms at corresponding positions),  $A_1$  and  $\neg A_2$  (or  $\neg A_1$  and  $A_2$ ) are referred to as *complementary* literals. We denote complementary literals as  $L_1 \neq L_2$  where  $L_1$  and  $L_2$  are an atom and its negation. For instance,

$\text{Resource\_agent}(\text{agent1}) \neq \neg \text{Resource\_agent}(\text{agent1})$

**Definition 2.** Given two literals  $L_1$  and  $L_2$  that are syntactically different and semantically opposite of each other (the assertion of  $L_1$  ( $L_2$ ) implies the negation of the other  $L_2$  ( $L_1$ )), we call  $L_1$  and  $L_2$  as *mutually exclusive* literals and use  $L_1 \neq L_2$  to denote it. For example,

$\text{Animal}(\text{seaCucumber}) \neq \text{Vegetable}(\text{seaCucumber})$

For two literals  $L_1$  and  $L_2$  that are syntactically different, but logically equivalent, we call them *synonymous*, and denoted  $L_1 \equiv L_2$ .

**Definition 3.** Given  $L_1$  and  $L_2$  that are a complementary pair of synonymous literals, we call them *incompatible* literals, and denote  $L_1 \not\equiv L_2$ . For example,

$\text{Father}(x, \text{john}) \not\equiv \neg \text{Male\_parent}(x, \text{john})$

For two literals (or concepts as in description logics)  $L_1$  and  $L_2$ , if  $L_1$  is a *subtype* or a *specialization* of  $L_2$ , then we say that  $L_1$  is subsumed by  $L_2$  (or  $L_2$  subsumes  $L_1$ ) and use  $L_1 \sqsubseteq L_2$  to denote that.

**Definition 4.** If  $P_1 \sqsubseteq P_2$  belongs to the taxonomy of concepts in a domain but a requirement description yields  $P_1 \sqsubseteq \neg P_2$ , then we call  $P_1$  ( $L_1$ ) and  $\neg P_2$  ( $L_2$ ) *anti-subtype* literals and use the following to denote it:  $L_1 \not\sqsubseteq L_2$ . For example,

$\text{Surgeon}(\text{john}) \not\sqsubseteq \neg \text{Doctor}(\text{john})$

If  $L$  is a *supertype* consisting of a list of subtypes denoted as  $\sqcup L_i$ , we use  $L \Leftrightarrow \sqcup L_i$  to represent the fact that  $L$  corresponds to the subtypes in  $\sqcup L_i$ .

**Definition 5.** When a description uses  $\sqcup L_i'$  for a supertype  $L$  that is defined by  $L \Leftrightarrow \sqcup L_i$ , if  $(\sqcup L_i') \not\equiv (\sqcup L_i)$ , we say that  $(\sqcup L_i')$  is *anti-supertype* with regard to  $L$  and use  $L \not\Leftrightarrow (\sqcup L_i')$  to denote that.

For instance, the supertype *Agent* in a particular organization consists of the following subtypes: user

agents, broker agents, resource agents and ontology agents.

$\text{Agent}(x) \Leftrightarrow (\sqcup L_i)$  where  $\sqcup L_i$  contains:

$[\text{User\_agent}(x) \vee \text{Broker\_agent}(x) \vee$

$\text{Resource\_agent}(x) \vee \text{Ontology\_agent}(x)]$

Let  $\sqcup L_i'$  be  $[\text{User\_agent}(x) \vee \text{Broker\_agent}(x)]$ , then

$\text{Agent}(x) \not\Leftrightarrow (\sqcup L_i')$

Given two literals  $L_1$  and  $L_2$  that share the same predicate, if the predicate in  $L_1$  and  $L_2$  is a symmetric relation (i.e., if  $L_1$  is  $p(x, y)$  and  $L_2$  is  $p(y, x)$ ), then

$\forall x \forall y [p(x, y) \supset p(y, x)]$

$L_1$  and  $L_2$  are referred to as *symmetric* and are denoted as  $L_1 \psi L_2$ .

**Definition 6.** When the predicate in  $L_1$  and  $L_2$  is a symmetric relation, but  $L_1$  and  $L_2$  are no longer symmetric literals, we say that  $L_1$  and  $L_2$  are *asymmetric* and use  $L_1 \not\psi L_2$  to denote that. Following is an example of asymmetric literals (assuming that the “connected” relation is only discussed with regard to agent1 and agent2):

$\text{Connected}(\text{agent1}, \text{agent2}) \not\psi \text{Connected}(\text{agent2}, \text{agent3})$

When two predicates represent relationships that are opposite of each other, we call them *inverse* predicates. Literals  $L_1$  and  $L_2$ , are referred to as inverse literals when they contain inverse predicates and represent opposite relationships. We use  $L_1 \neq L_2$  to denote that.

**Definition 7.** When predicates in  $L_1$  and  $L_2$  represent inverse relationships but  $L_1$  and  $L_2$  are no longer inverse literals, we say that  $L_1$  and  $L_2$  are *anti-inverse* and denote it with  $L_1 \neq L_2$ . For instance,

$\text{Send\_msg\_to}(\text{agent1}, \text{agent2}) \neq$

$\text{Received\_msg\_from}(\text{agent2}, \text{agent3})$

When a compound predicate ( $L$ ) is fully defined through a logical expression of other predicates  $(\sqcap L_i)$ , we use  $L \equiv \sqcap L_i$  to denote it.

**Definition 8.** When a logical expression  $\sqcap L_i'$  is used in a model for the compound predicate  $L$  that is fully defined by  $L \equiv \sqcap L_i$ , if  $(\sqcap L_i') \not\equiv (\sqcap L_i)$ , we say that  $L$  and  $(\sqcap L_i')$  are *mismatching* and use  $L \not\equiv (\sqcap L_i')$  to denote that.

$\text{Mobile\_agent}(\text{agent1}) \equiv [\text{Executing}(\text{agent1}, \text{host1}) \wedge$

$\text{Executing}(\text{agent1}, \text{host2}) \wedge \text{host1} \neq \text{host2}]$

$\text{Mobile\_agent}(\text{agent1}) \not\equiv [\text{Executing}(\text{agent1}, \text{host1})]$

Given  $L_1$  and  $L_2$  for the same proposition and  $L_2$  is at a more concrete level of abstraction than  $L_1$ , we call them *reified* literals and denote it with  $L_1 \cong L_2$ .

**Definition 9.** If reified quantities in  $L_1$  and  $L_2$  are no longer compatible, we say that  $L_1$  and  $L_2$  are *disagreeing* and use  $L_1 \not\cong L_2$  to denote it. For example,

$\text{Memory\_capacity}(\text{agent1}, 2\text{GB}) \cong$

$\text{Memory\_capacity}(\text{agent1}, 1500\text{MB})$

**Definition 10.** Given literals  $L_1$  and  $L_2$  with either the same or different predicate symbols, if they contain attributes (terms) which violate type restrictions or integrity constraints, we refer to  $L_1$  and  $L_2$  as *contradictory* and denote it with  $L_1 \neq L_2$ .

Created(agent1, 11-1-2007)  $\neq$   
 In\_service(agent1, 10-1-2007)

When our beliefs about the world are not of a total commitment, we say that the beliefs are *uncertain*. In many real world circumstances, we have to deal with the issues of how to represent the strength of an uncertain belief and reason about uncertain beliefs during software development process. This is quite different from nonmonotonic reasoning [9]. Inconsistency can crop up in uncertain belief representation and reasoning.

In probabilistic logic, there is an important issue that pertains to the *probabilistic entailment* [9]: given a set  $\Delta$  of sentences and their probabilities, determine the probability of a sentence  $\alpha$  that logically follows from  $\Delta$ . For instance, given  $\Delta = \{A_1, A_1 \supset A_2\}$  and the probabilities:  $\text{Prob}(A_1)$  and  $\text{Prob}(A_1 \supset A_2)$ , we would like to establish a consistent  $\text{Prob}(A_2)$ . For the given sentences, there are four possible interpretations as given below where each column corresponds to a possible interpretation:

$A_1$	true	true	false	false
$A_1 \supset A_2$	true	false	true	true
$A_2$	true	false	true	false

The consistent truth value assignments are shown below in the matrix  $V$  where 1 represents “true” and 0 denotes “false”.

$$V = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

The probability values for the three sentences are constrained by the following matrix equation [9]:

$$\Pi = VP$$

and by the probability axioms of  $\sum_i \text{Prob}_i = 1$  and  $0 \leq \text{Prob}_i \leq 1$  for all  $i$ .

A geometric interpretation is given in [9] for the aforementioned constraints. There is a convex region of consistent probability values specified by the extreme values of  $\Pi$ . Thus a consistent value for  $\text{Prob}(A_2)$  must be chosen from within the convex hull of the region.

**Definition 11.** Given a set  $\Delta$  of sentences and their probabilities, and a literal  $L$  that logically follows from  $\Delta$ , if  $\text{Prob}(L)$  is outside the convex region specified by the extreme values of probabilities for  $\Delta$  and  $L$ , then we say that the assignment of the probabilistic truth value for  $L$  is inconsistent. We call this type of inconsistency as *iProbVal* (for inconsistent probabilistic value) and use  $\text{Prob}(L) \notin \text{Prob}(\Delta)$  to denote it.

#### 4. Managing Inconsistency in VBSE

Managing inconsistency in VBSE does not have to start from scratch. We can still define a set of consistency rules that describe proper relationships that must be maintained among VBSE descriptions. However, existing

results on inconsistency management obtained from value-neutral software development should be augmented in several directions.

First, consistency rules used for detecting and handling inconsistencies need to be extended according to the following.

- There should be consistency rules defined for the most frequently encountered model conflicts among stakeholders’ success models. Listed in [4] are some of the most common ones. SVPs can be a major source of inconsistency as stakeholders tend to have their value propositions that are emergent and are not necessarily compatible with each other.
- Since identifying all of the success-critical stakeholders (SCS) is a necessary condition for the SCS win-win achievement [3], there should be consistency rules that account for all the SCSs.
- In the WinWin Negotiation model in [3], there are agreements that cover win-conditions from stakeholders. The options adopted by an agreement address some divergent issues that involve win-conditions from certain stakeholders. For instance, if stakeholder A’s win-condition is to select the Windows platform, whereas stakeholder B’s win-condition is to select the Unix platform, then the two conditions are conflicting. If the two agree to have a middle ground of running the system on Java Virtual Machine (JVM), then we have an agreement that adopts the JVM option for the platform issue that involves the win-conditions from the two stakeholders. The model will reach an equilibrium state when all win-conditions are covered by agreements and there is no outstanding issue left [3]. For the model to work, there need to be consistency rules that recognize all the inconsistent circumstances among win-conditions.

Secondly, resolutions to inconsistencies stemming from either conflicting SVPs or divergent business objectives should be based on a broader range of approaches. As delineated in [3], there are utility theory, dependency theory, decision theory, and control theory serving as the theoretical underpinnings of VBSE. When resolving a conflict description, we can resort to various tenets of decision theory: negotiation theory, game theory, multi-attribute decision theory, statistical decision theory, real options theory, and the Theory of Justice [3].

Third, additional inconsistency detection method can be defined through techniques based on trace dependencies. To ensure that a system’s objectives and its value considerations are embodied in the software’s design and development practices, the software traceability techniques play an important role in VBSE [2]. During the software development process, many artifacts are produced and maintained: documents, requirements, design models, test scenarios, and so forth.

Trace dependencies are to identify relationships among those artifacts and the quality of the trace dependencies should reflect the value of the artifacts they attempt to bridge. This can be utilized for detecting inconsistency as well as for documentation, program understanding, impact analysis, reuse, quality assurance, user acceptance, error reduction, cost estimation, and customer satisfaction.

Finally, value-based risk management can be used for assessing the impact and risk of unresolved inconsistencies.

## 5. Conclusion

VBSE offers a new software development paradigm that recognizes the importance of business and stakeholders value considerations. It tackles the decision making process in software development and maintenance from a value-based perspective. However, inconsistencies that arise during software development pose direct threats toward the value-based objective, if they are not properly tamed. In this paper, we reviewed the current landscape of inconsistency management in the value-neutral software engineering setting. After examining the dimensions of inconsistency as a phenomenon in software development process, we propose some guidelines on how to tame conflicting descriptions and inconsistencies in value-based software engineering. The take-home message is that inconsistency is an issue to be reckoned with in value-based software engineering.

Future work can be pursued in the following directions. Specific consistency rules can be defined for frequently encountered stakeholders' divergent success models. An equally important set of consistency rules is also in order for the WinWin Negotiation model.

## Acknowledgements

The author would like to thank anonymous reviewers for their valuable comments which help improve the paper's presentation.

## References

1. S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grunbacher (ed.), *Value-Based Software Engineering*, Springer, Berlin, 2006.
2. B. Boehm, "Value-Based Software Engineering: Overview and Agenda," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
3. B. Boehm and A. Jain, "An Initial Theory of Value-Based Software Engineering," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
4. B. Boehm, "Value-Based Software Engineering: Seven Key Elements and Ethical Considerations," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
5. R. J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, San Francisco, 2004.
6. F. Brooks, "No Silver Bullet: Essence and Accident in Software Engineering," *IEEE Computer*, Vol. 20, No. 4, 1987, pp.10-19.
7. G. Cugola, E. Di Nitto, A. Fuggetta, and C. Ghezzi, "A Framework for Formalizing Inconsistencies and Deviations in Human-Centered Systems," *ACM Transactions on Software Engineering and Methodology*, Vol. 5, No. 3, July 1996, pp. 191-230.
8. H. Erdogmus, J. Favaro, and M. Halling, "Valuation of Software Initiatives under Uncertainty: Concepts, Issues, and Techniques," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
9. M. R. Genesereth, and N. J. Nilsson, *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Los Altos, CA, 1987.
10. P. Grunbacher, S. Koszegi and S. Biffl, "Stakeholder Value Proposition Elicitation and Reconciliation," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
11. J. Grundy, J. Hosking, and W. B. Mugridge, "Inconsistency Management for Multiple-View Software Development Environments," *IEEE Transactions on Software Engineering*, Vol. 24, No. 11, November 1998, pp. 960-981.
12. A. Hunter and B. Nuseibeh, "Managing Inconsistent Specifications: Reasoning, Analysis, and Action," *ACM Transactions on Software Engineering and Methodology*, Vol. 7, No. 4, October 1998, pp. 335-367.
13. B. Nuseibeh, S. Easterbrook, and A. Russo, "Leveraging Inconsistency in Software Development," *IEEE Computer*, Vol. 33, No. 4, April 2000, pp. 24-29.
14. W. N. Robinson and S. D. Pawlowski, "Managing Requirements Inconsistency with Development Goal Monitors," *IEEE Transactions on Software Engineering*, Vol. 25, No. 6, 1999, pp. 816-835.
15. I. Sommerville, P. Sawyer, and S. Viller, "Managing Process Inconsistency Using Viewpoints," *IEEE Transactions on Software Engineering*, Vol. 25, No. 6, 1999, pp. 784-799.
16. G. Spanoudakis and A. Zisman, "Inconsistency Management in Software Engineering: Survey and Open Research Issues," in *Handbook of Software Engineering and Knowledge Engineering*, S.K. Chang (ed.), World Scientific Publisher, 2001, pp.329-380.
17. A. van Lamsweerde, R. Darimont, and E. Letier, "Managing Conflicts in Goal-Driven Requirements Engineering," *IEEE Transactions on Software Engineering*, Vol. 24, No. 11, November 1998, pp. 908-926.
18. C. Wohlin and A. Aurum, "Criteria for Selecting Software Requirements to Create Product Value: An Industrial Empirical Study," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
19. H. Zhu, and L. Shan, "Well-Formedness, Consistency and Completeness of Graphic Models," Proc. of the 9<sup>th</sup> International Conference on Computer Modeling and Simulation (UKSIM 2006), Oxford, UK, April 4-6, 2006. pp. 47-53.



# WSTester: Testing Web Service for Behavior Conformance

Bixin Li<sup>1,2</sup>, Lili Yang<sup>1</sup>, Shunhui Ji<sup>1</sup>, Dong Qiu<sup>1</sup>, and Xufang Gong<sup>1</sup>

<sup>1</sup>School of Computer Science and Engineering, Southeast University

Nanjing 210096, Jiangsu Province, P.R.China. Email: bx.li@seu.edu.cn

<sup>2</sup>Department of Computer Science and Engineering, University of California  
Riverside, CA92521, USA. Email: lbxin@cs.ucr.edu

## Abstract

*In this paper, WSTester is introduced simply to show how to test behavior conformance between Web services based on interaction behavior specification and extended Labeled Transition System, corresponding experiment results analysis show the significance of the tool.*

## 1 Introduction

In Web service times, it is necessary to realize the precise interaction and rapid integration of heterogeneous applications. Old independent point-to-point solution is unable to satisfy such new application requirement, it must be replaced with service-guided distributed computing architecture. SOA (Service-Oriented Architecture) is a new paradigm that can be used to satisfy user's current requirement. As a typical application case of SOA, Web service has won the wide support from academia and industries. Web service is a network component over open software platform, it supports interactive operation between different machines connected by the internet, it inherits the merit of XML language, it adapts and supports international open technology standards and specifications, where WSDL (Web Service Definition Language) is used to describe Web service, UDDI (Universal Description, Discovery, and Integration) is responsible for publishing and registering Web service in a *Register Center* so that it is easy for service requestor to find his wanted service, SOAP (Simple Object Access Protocol) protocol is used to bind and call it after a service is found [1].

Since some ideas of Web service are originated from object-orientation and component technology, they have some common features. However, the obvious difference between them is that a COTS component or an object is physically integrated into application system developed by the users, while only functions of Web services can be used in their application by remote calling, the real running of a

service body is performed in the server located on the end of service provider. Those services interacting each other in an application are in fact distributed in different organizations or departments. This reason makes it more complicated and difficult to test the interactive behavior of Web service.

Next, WSTester is discussed about how to test behavior conformance from the user's viewpoint based on an extended Labeled Transition System called xLTS discussed in [2], Labeled Transition System in [3], and *interaction behavior specification* introduced in [2], which was enlightened by the idea in both [4] and [5].

## 2 WSTester

WSTester(Web Services Tester) is a conformance testing experimental tool, which integrates xLTS model and UML sequence diagram with OCL constraints. WSTester also provides a set of tools for supporting specification analysis, model transformation and model-based testing, it also provides an exchangeable format for integrating with other UML tool since its interface is based on XMI. WSTester has functional components for generating test case and executing test process automatically, where xLTS model which is transferred from UML model is the solid base for generating test case.

The flowchart of WSTester is described in Figure 1, which includes four parts:

- *Generate formal behavioral model:* based on UML 2.0 sequence diagram and OCL constraint, LTS is extended to be xLTS with semantic information, both data flow and control flow information can be captured in xLTS and more rich information can be provided by xLTS to generate test case.
- *Generate test sequences and test cases:* based on xLTS, enough and wanted test sequences and test cases are obtained.

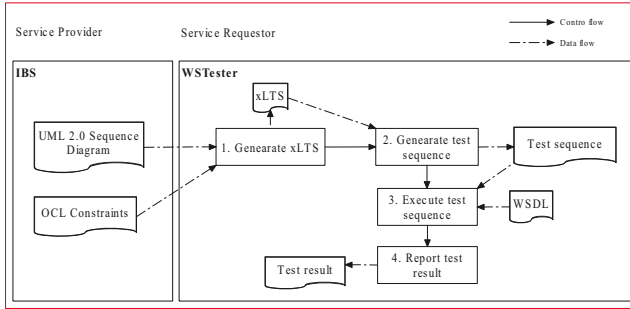


Figure 1. Flow chart of WSTester

- *Execute test sequence*: use test cases to test service by remote calling and executing related services.
- *Output test result*: test report is output based on test process record and final test result.

WSTester has four functional components: *xLTS model transformer*, *Test case generator*, *Main tester* and *wsCaller*, their main functions are introduced as follows:

- (1) *xLTS model transformer* will be used to transfer interaction behavior specification into *xLTS* so as to generate wanted test case to satisfy test requirement.
- (2) Test case generator is used to generate test case based on *xLTS*.
- (3) Dominant tester is the center modular for controlling test execution process that determines which test actions will be taken.
- (4) *wsCaller* is used to call service to be tested.

### 3 CSW: an Illustration Example

Let's see a sample service CSW (Customer-Supplier-Warehouse), which is borrowed from [6]. In the CSW service, the expected behaviors will be described as follows:

- (1) Customer sends Supplier a request message *requestQuote* for inquiring the goods about quote price information;
- (2) Supplier returns a response message *requestQuote.r* to reply the request
- (3) Customer accepts the quote price and sends a message *orderGoods* to Supplier for ordering the goods;
- (4) Supplier will send a message *checkShipment* to Warehouse after he accepts the order form;
- (5) Warehouse checks the repertory to check whether it is all right to consignment or not;
- (6) Supplier will make different decisions according to the checking result to Warehouse:

- (6a) If it is no problem to shipment now, Supplier will send an acknowledgement message to Customer

SN...	Implementation
1	correct implementation
2	when inputed quantity of goods is 0, quote of goods can be requested
3	quote information of a non-existence is requested
4	there is inconsistency between requested goods and returned goods
5	amount of requested goods is less than amount of returned goods
6	order form number generated automatically is inconsistent with ordering process
7	amount of requested goods is beyond biggest repertory, but it is ordered
8	order form numbers generated in different times are not identified
9	order is recored in system even it has been cancelled
10	payment amount is computed incorrectly

Figure 2. Different implementation of CSW

for telling him that the order form has been accepted, and Customer should do a *makePayment* operation to make a payment, then Supplier asks Warehouse to shipment, Warehouse sends Customer the message *getShipmentDetail* to ask Customer the shipment detail, Customer sends Warehouse the message *confirmShipment* to confirm this shipment, finally Warehouse sends Supplier a message to confirm the message *confirmShipment*.

- (6b) If it is impossible to shipment now, Supplier sends Customer a message *cancelOrder* to cancel order form.

## 4 Experiment Result Analysis

For validating *xLTS* based method, one kind of correct implementation and nine kinds of error implementations of sample example CSW are designed in Figure 2 to check the ability of our method and WSTester. For each implementation, we generate test case from LTS and *xLTS* model respectively and observe what differences will happen when each of them is used independently.

### 4.1 Evaluation factors

Two important factors needed to be checked to determine the ability of a test method: *error-checking capability* and *test expensive*. Error-checking capability is usually measured using *test coverage rate* (or TCR), while test expensive (or TE) is measured by the *length of test sequence* (or LOT). Our test goal is to perform a test with smallest test expensive and strongest test capability. However the two aspects are usually contradictory each other, so we pursuit a strongest test capability when the contradiction can not be solved. In our method, test coverage rate is computed using the number of checked out errors (NCE) and the number of total errors (NTE):

$$TCR = \frac{NCE}{NTE}$$

$$TE = LOT$$

SN	Test case	LOT
1	$\underline{m}$ pass[] otherwise; fail	0
2	?requestQuote;(!requestQuote; ( $\underline{m}$ ; pass[] otherwise; fail))[]otherwise; fail))	2
3	?requestQuote;!requestQuote;?orderGoods;!confirmOrder; ( $\underline{m}$ pass[]otherwise; fail) []otherwise; fail)	4
4	?requestQuote;!requestQuote;?orderGoods;!cancelOrder; ( $\underline{m}$ pass[]otherwise; fail) []otherwise; fail)	4
5	?requestQuote;!requestQuote;?orderGoods!confirmOrder;?makePayment; ( $\underline{m}$ ; pass[]otherwise; fail)	5

Figure 3. LTS-based test case

SN	Test results	
	TCR (%)	Test conclusion
1	0.0	Pass
2	0.0	Pass
3	60.0	Fail
4	80.0	Fail
5	0.0	Pass
6	60.0	Fail
7	60.0	Fail
8	0.0	Pass
9	0.0	Pass
10	20.0	Fail

Figure 4. Test conclusion

## 4.2 LTS-based test

At first, let's see what will happen when we use LTS-based test method proposed by Jiang [3], supposing the biggest loop times is 1, then five test cases are generated from xLTS in Figure 3, where  $\underline{m}$  represents empty message  $\theta$ . The five test cases are used to test 10 kinds of different implementations of CSW service, and the test result is given in Figure 4.

We can see from Figure 4, only five kinds of error implementations of CSW service have been discovered, which is caused by the construction of test execution trace with sequence dependence relation. Because control-flow information is considered in LTS-based method, it is easy to check out the error produced in sequence operation. However, for LTS-based method, it is needed to add data into test case manually, which limits the test capability of LTS based method, added data will affect directly whether more errors can be checked out or not. The reason for both the 2nd and 5th error haven't been checked out is that LTS is short of related data-flow information. It is needed to declare that it is our limitation to loop times, which causes both 8th and 9th error haven't been checked out.

SN	Test case	LOT
1	?requestQuote<prodA,500>!requestQuote<1,prodA,490,10.0,4900>?OrderGoods<1,400>!confirmOrder<1>?makePayment<1,4000>	5
2	?requestQuote<prodA,0>!quiescence	2
3	?requestQuote<sun,400>!quiescence	2
4	?requestQuote<prodA,1>!requestQuote<1,prodA,400,10.0,4000>	2
5	?requestQuote<prodA,500>!requestQuote<1,prodA,490,10.0,4900>?OrderGoods<1,400>!confirmOrder<1>?makePatment<2,4000>	5
6	?requestQuote<prodA,500>!requestQuote<1,prodA,490,10.0,4900>?OrderGoods<1,500>!quiescence	4
7	?requestQuote<prodA,500>!requestQuote<1,prodA,490,10.0,4900>?OrderGoods<1,400>!confirmOrder<1>?makePayment<1,4000>?requestQuote<prodB,500>!requestQuote<2,prodB,300,20.0,6000>?OrderGoods<1,300>!confirmOrder<1>?makePayment<2,6000>	10
8	?requestQuote<prodA,500>!requestQuote<1,prodA,490,10.0,4900>?OrderGoods<1,400>?cancelOrder<1>?requestQuote<prodA,500>!requestQuote<2,prodA,490,10.0,4900>	6
9	?requestQuote<prodA,500>!requestQuote<1,prodA,490,10.0,4900>?OrderGoods<1,400>!confirmOrder<1>?makePayment<1,4000>	5

Figure 5. xLTS-based test case

SN	Real results	
	TCR (%)	Test conclusion
1	0.0	Pass
2	11.1	Fail
3	11.1	Fail
4	77.8	Fail
5	77.8	Fail
6	44.4	Fail
7	11.1	Fail
8	11.1	Fail
9	11.1	Fail
10	44.4	Fail

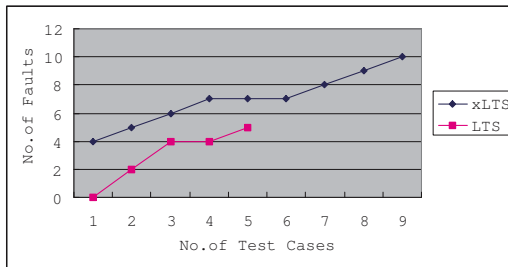
Figure 6. Test conclusion

## 4.3 xLTS-based test

Now we will observe what will happen when we use our xLTS-based test method. Test cases are listed in Figure 5, where test sequences with different lengths are adopted, and requests with different amounts for requesting quote are input.

As we can see from Figure 6, all the errors included in the nine kinds of incorrect implementation of CSW service have been checked out. But the discovering of most of them is based on the construction of sequence dependence test execution trace, such as 6th, 8th and 9th errors. It is very hard to check out them to use current methods based on single service operation. In error checking, the check capability of a tool is affected by length of test trace obviously. In above example, some errors are easy to find by checking quote information of goods. For example, the error of inconsistency between requested goods and returned goods is checked out easily by checking a trace with length equaling to 2: observing the response after the quote information is requested.

However, there are a lot of errors that can not be found so



**Figure 7. Test comparison**

easily, they can only be found in sequence execution process of multiple operations. For example, to check out the error that the numbers of two order forms of goods are the same for twice successful advance booking in different time, it is needed to run two complete advance booking process where the length of trace is 10. This shows, for the error existing in sequence operation process, the longer the test trace is, the stronger test capability is, but the test expensive for generating test case is increasing accordingly. To overcome this shortcomings, we borrow on-the-fly test strategy, i.e., test action is accompanying with test case generation, stopping the test process as soon as the error is found, instead of starting test after all of test cases have been generated, which ignores the space explosion problem caused by model-driven approach and reduces test expensive.

We can see from Figure 7 that the error checking coverage rate of xLTS-based method is higher than that of LTS-based method, the main reason is that data-flow information has been included in former method. But it is needed to point that coverage rate 100% doesn't mean that our method has 100% coverage for any test. It is no meaning for coverage rate itself, but it can be used as a reference data when we do a compare between two methods.

## 5 Conclusion

There are some people who are doing research on behavior conformance testing, such as [5], [7] and [8]. Enlightened by the main idea in these related work, both *interaction behavior specification* and xLTS are introduced to generate test case for testing behavior conformance in this paper and its earlier version [2]. However, there are still some shortcomings with our method, which encourages us to do further work in our future time. Two representative issues are summarized as follows: (1) the inconsistency caused by synthesizing xLTS from sequence diagrams must be checked and solved, because the xLTS model generated automatically from specification is just an approximation of system, the prototype tool must support user's manual modification; (2) the way for combining xLTS model from

many different sequence diagrams must be considered in next work. Because different sequence diagrams may have some same or similar behaviors, how to combine and confirm these behaviors is a challenge problem.

## Acknowledgements

The authors thank Prof. Rajiv Gupta in University of California Riverside for providing a very comfortable Lab. This work is partially supported by the National Nature Science Foundation of China under No.60773105, partially by the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2007513, and partially by National High Technology Research and Development Program under Grant No. 2008AA01Z113.

## References

- [1] W3C. *Web Services Activity*. <http://www.w3.org/2002/ws/>.
- [2] B. Li, X. Fan, and L. Yang. *Extending Labeled Transition Systems for Conformance Testing of Web Services*. Technical Report, Southeast University, 2009.
- [3] F. Jiang, Z. Ning. *Automatic Test Case Generation Based on Labeled Transition System*. Chinese Journal of Computer Research and Development, 2001, vol 38, no. 12.
- [4] S. Pickin, C. Jard, T. Jeron, J. Jezequel, and Y. Traon. *Test Synthesis from UML Models of Distributed Software*. IEEE Transaction on software engineering, April 2007, vol 33, no.4.
- [5] E. Cartaxo, F. Neto, and P. Machado. *Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems*. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 7-10 Oct. 2007.
- [6] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architecture and Applications*. Springer Verlag, 2004. ISBN 3-540-44008.
- [7] J. Tretmans. *Conformance testing with labeled transition systems: Implementation relations and test generation*. Computer Networks and ISDN Systems, 1996, 29:49-79.
- [8] R. Heckel, L. Mariani. *Automatic conformance testing of web services*. In: Proceedings of FASE, Edinburgh, Scotland, Apr., 2005, 2-10.

# Robustness Verification Challenges in Automotive Telematics Software

Ali Shahrokni, Robert Feldt

Computer Science and Engineering  
Chalmers University of Technology  
Gothenburg, Sweden

Fredrik Petterson, Anders Bäck

Volvo Technology AB  
Gothenburg, Sweden

**Abstract**—The automotive industry has always had a strong pressure of ensuring that only high quality software is allowed to control the vehicle. The general increase in the amount of software in a modern vehicle and trends in the industry is creating more open standards and systems. In particular, the software architectures used will have to support extension with 3<sup>rd</sup> party components and extension at run-time during normal operation of the vehicle. These trends put additional pressure on the automotive industry to verify and validate the quality of the systems. Since the software often governs safety-critical features there are high demands on the robustness of the final system. It is currently not clear how these robustness testing challenges should be met and previous research have not addressed this fully. We outline the challenges and point to possible solutions that the research community needs to work together with the automotive industry to realize.

**Keywords**—software testing; Robustness testing; verification and validation; automotive industry; open systems; run-time extension;

## I. INTRODUCTION

Robustness is a non-functional property of software systems. Non-functional requirements (NFR) or properties have been widely discussed in the automotive industry for several years. Verification of NFRs is a costly process and great challenge in many parts of the automotive industry. A key factor creating this challenge is the fact that many existing testing and verification solutions are not usable in the industry. Another factor is the difficulty to specify testable and measurable NFRs. Since requirements and verification are deeply interconnected this has caused difficulties for verification of NFRs in some areas.

Telematics or vehicle communication systems have grown rapidly in the previous years. A trend that has been and will be given much attention in the automotive telematics industry is the movement of the systems towards more open standards and architectures.

In this paper we identify trends and goals in the automotive telematics industry when moving towards open systems. Furthermore, we describe the state-of-practice of robustness and dependability verification in the automotive telematics industry. Considering the trends, we have identified and discussed what future challenges exist in the field of robustness and dependability verification in the field. Comparing the challenges with the existing solutions found in the state-of-art and state-of-practice in other fields we

have suggested ways on how to face and tackle these problems.

## II. BACKGROUND

In this chapter some of the concepts and trends that will be discussed in this paper are presented and defined.

### A. Dependability and Robustness Definition

Dependability is an ‘umbrella’, ‘integrative’ concept having multiple attributes [1]. Formally it and its basic sub-concepts are defined as [2]:

“Dependability of a computing system is the ability to deliver service that can justifiably be trusted. The service delivered by a system is its behavior as it is perceived by its user(s);”...”Correct service is delivered when the service implements the system function. A system failure is an event that occurs when the delivered service deviates from correct service.”

Robustness is informally defined as dependability with respect to erroneous input [2]. However it is clear that it is not considered a main attribute of dependability, but is characterized as a secondary and specializing attribute [2]; I.e. dependability with respect to external faults which characterizes a system reaction to a specific class of faults.

In older texts robustness has been defined as  $\{, \#9\}$ :

“The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.”

### B. Goals and trends in telematics software systems in the automotive industry

Telematics is typically defined as a combination of telecommunication and computing, i.e. data communication between systems and devices. The term automotive telematics refers to information-intensive applications combining telecommunications and computing technology in vehicles [4]. Often it is implied that such systems also include a Global Positioning System (GPS) unit.

Telematics software or telematics software system (TSS) refers to a piece of software that is used in a telematics system. A TSS can be used for many different purposes, including: managing road usage and collecting road tolls, pricing auto insurance, tracking fleet vehicle location and logistics (fleet telematics), car accident prevention, remote diagnostics to ensure uptime, infotainment applications etc.

Some articles outline general challenges for software development and engineering in the automotive application domain [5, 6]. Like in many other industrial sectors, software is increasingly driving innovation and product development

in the automotive industry. The trend that the share of software in vehicles is steadily rising (some even report it is rising exponentially [6]) is accompanied by general trends like rising time and cost pressure and increasing quality demands. DaimlerChrysler experts in 2003, estimated that 80% of future innovation will be driven by electronics and 90% of that by software [5]. The software is also getting more complex; instead of isolated systems they now have to interoperate and communicate with the outside world.

In particular for telematics, the development cycles are even shorter than for product development in general [5]. This calls for open and flexible architectures. The vehicles should support dynamic extensions that might not have been developed in-house but by 3<sup>rd</sup> party suppliers. For example, the core software architecture will need to provide telematics services. These services can be used by plug-in or third party components such as vehicle logging and drive management.

The need and will to standardize software components and use existing commercial software in vehicles is a future trend in the automotive industry. This trend puts great responsibility on vehicle companies to provide a robust platform for this kind of software. The host platform should make sure that the third party components are compatible with the rest of the software system and do not interrupt the function of the system. Furthermore it is important to make sure that the quality attributes of the software like availability, dependability and robustness are intact and not worsened due to the existence of the third party component.

A common view is that future telematics systems will be a single computing platform that offers many different applications and services [7].

### C. Open standards and architectures

Due to the rising complexity of software systems there is a need to standardize systems and architectures. This challenge is sensed significantly in automotive industry. This standardization will help decrease the development and customization costs for new systems. It will help integrating different solutions from different automotive companies much simpler with more clear communication interfaces. An example of this is the Autosar [8] standard which is rapidly becoming a key technology in the next generation of vehicles.

There have been attempts to create standards for telematics systems in the recent years. CVIS [9] is one of the projects trying to establish a standard for the future vehicle telematics systems. CVIS was started by the Information Society and Media directorate general in European commission. The goal of the project is to develop intelligent co-operative systems based on vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communication. An important purpose of such integration is to increase the safety and efficiency on roads. Other purposes of CVIS project are to increase road network capacity, reduce congestion and pollution, improve traffic safety for all road users, improve efficiency of logistics and response to hazards, incidents and accidents.

CVIS in its turn is based on other open platforms like OSGI [10] to achieve a better grade of standardization and reusability.

OSGi is a dynamic module system for Java that acts as a generally useful middleware [10]. OSGi technology provides a platform for service delivery based on components. It is used in several different application areas and domains such as IDEs (Eclipse), application servers (IBM Websphere, JBoss), industrial automation, mobile phones etc [10].

The claimed benefits of OSGi technology are reduced complexity by the use of bundles (the OSGi components), reuse of 3rd party components, dynamic update of bundles and services, easier deployment by specified install and bundle management, adaptive mixing and matching of components, and a useable security model that extends the one supplied by Java [10]. The core API is a single Java package comprised of 30 classes/interfaces.

An OSGi bundle is a Java JAR file where all things that are not explicitly exported by the developer are hidden. Similarly a user of a bundle must explicitly import the parts they need.

A bundle can register services (embodied in Java objects) in the OSGi service registry. It can also get a service and listen for a service to appear or disappear. A filter language can be used to ensure that the proper services are detected. The filtering is based on the properties of the service. Services can be dynamically withdrawn, added or changed. Bundles are deployed on an OSGi framework supplying a runtime environment for executing bundles. Bundles run in the same Java VM and can share code.

There are several open-source implementations of OSGi frameworks, such as Knopflerfish and Newton.

### III. STATE OF AUTOMOTIVE TELEMATICS VERIFICATION PRACTICE

The automotive industry is increasingly dependent on software and needs to develop more effective and efficient methods to ensure the quality of software. These needs are even more evident given recent trends towards more open standards, systems and architectures[5, 6]. Given these facts, creating useful methods for testing is very attractive and cost saving.

During the course of this project an extensive literature study was performed. The purpose was to identify the state of the art of verification and validation of non-functional attributes of software systems. A complete summary of the results found is out of the scope for the present paper.

In summary, the results from the review show that several tools and methodologies on verification of robustness and dependability have been presented [11-14]. However, none of these tools are designed for end-to-end testing of communications systems used in the telematics industry. This has made them less interesting for the telematics industry to use. More information about these tools is given later in this paper.

Furthermore telematics have been a part of the automotive industry only for a few years and is only in the

early stages of its evolution. Many unexplored opportunities and applications have been recently pointed out [9].

#### A. State of Practice

To get more information on the state of practice in verification of automotive telematics systems we conducted six semi-structured interviews with practitioners. The goal was to identify the state of practice in verification of NFRs in telematics systems at Volvo Technology. Another goal was to recognize the trends and challenges in eliciting and verifying NFRs of these systems. The interviewees were all working on telematics related projects at Volvo 3P and Volvo Technology in Gothenburg at the time of the interviews. In this paper these companies will be collectively referred to as Volvo. Whenever Volvo is mentioned it refers to the telematics parts of these two companies.

Two more interviews were conducted in another company. This company will be referred to as company 2 in this paper. Company 2 develops safety-critical software systems in a different industrial area which is more mature in regards to dependability and non-functional properties. Thus, our goal is to contrast their state-of-practice to the one at Volvo. Below we summarize the results from these different interviews; table 1 shows the number and types of roles involved.

#	Company	Role
1	Volvo Technology	System testing
2	Volvo 3P	Acceptance testing
3	Volvo Technology	Acceptance testing
4	Volvo Technology	System testing
5	Volvo Technology	Development
6	Volvo Technology	Requirement
7	Company 2	Project leader, development
8	Company 2	Product leader, system testing

Table 1. Roles of the interviewees

To have a better understanding of how the testing and verification process in general works at Volvo, interviewees were chosen from different parts and with different roles in the development process. The interviewees were asked to explain the general testing process and describe their parts in the process in more detail. Furthermore they were asked to explain how the verification of non-functional or quality requirements and more specifically robustness of the system works.

At Volvo the verification is mostly done through testing. The testing is done in the different phases of the system development.

The first phase is unit testing which is done by the developer. This test makes sure the functionality for that unit exists and functions properly.

After the system integration, system tests are conducted. These tests follow the requirement specification documents. Earlier the test process has been more scripted and specified. All the test cases were specified in detail and the tester had to follow them. However, a new working method has been introduced where less detailed instructions is given to the

tester which gives more freedom to the tester on how to test the functionality or the NFRs. In the end the testing is reviewed and necessary steps are taken if any part of the testing is wrong or incomplete.

The next step is to deliver the tested system to the customer which in this case is Volvo 3P. User acceptance test is run by Volvo 3P at this time. This part includes testing the system on the vehicle. The system will be installed in the vehicle and, while driving the vehicle, functionality will be verified in different situations. The tests are performed according to the user acceptance test specification which makes sure that the requirement specifications are fulfilled.

The testing in all of these phases is mainly done manually today. There are plans for using more automated testing tools in the future but most of the tools available do not fulfill the needs of the company. The main problem with these tools is their incapability to provide an end-to-end service over the telematics connection. In other words, a complete testing tool that can initiate the communication on one side and follow the process on the other side of the link is not available to the company.

According to the interviews, this fact makes manual testing still the cheapest, best and in many cases only way to verify the fulfillment of functional and non-functional requirements at Volvo.

Parts of the telematics system are however more developed in the area of non-functional testing. A good example of this is the communication channels. The telecommunication field is a mature field and there are more advanced automated testing tools and verification methods available for it. Using these tools the quality of the communication channel is tested with a relatively low cost.

With regards to NFRs of telematics systems the property receiving most attention is availability. Using the telecommunication testing tools mentioned above and simulators this property is tested. Performance is another property which is tangible and easy to understand. Therefore it is easier to consider and verify this property during the testing process than a more complex property like robustness. Additionally, since the system is currently closed for 3<sup>rd</sup> party developers and non-safety critical, validating properties like dependability and robustness is not highly prioritized.

At company 2 this looks different. Their systems are safety-critical. This makes robustness and stability essential properties of the system and they are considered to be even more important than availability. Although there is a high demand on availability it is less disastrous if the system is out of service than if it operates faulty and misleads the users. Since the customers have a higher demand on these properties, they are more clearly specified in the requirement specification than at Volvo. Thus they put in more effort to make the non-functional requirements measurable and testable. The systems should follow certain standards and should go through safety reviews before they are accepted by the customer.

Company 2 uses more automated testing tools on the unit testing level. However, on higher levels like integration and system tests the tests are still mostly done manually. They

use mathematical models and some testing tools for proving some of the NFRs like availability since these requirements are mostly specified with probability.

At Volvo, telematics systems are currently closed to other companies and entirely developed in-house. The system has clearly specified design and interfaces. There is a total control on the development and testing process and this makes the systems more robust and dependable. However, this fact is changing rapidly. The automotive industry and especially automotive telematics systems are clearly moving towards outsourcing parts of the development of their service applications to other companies. This results in more open systems. This means that other companies can develop software running on open platform provided by Volvo. This challenges the company to find better ways to validate robustness and dependability of their systems in the future. Looking into methods used by more mature companies in related fields, like Company 2, can be an important start. This matter is further discussed in the next chapter.

#### IV. CHALLENGES

Given the trends mentioned earlier, there are many challenges for robustness verification in the automotive telematics industry today. Below we discuss some of these challenges in more detail.

##### A. 3<sup>rd</sup> party components

According to the results from our interview study the introduction of components developed by 3<sup>rd</sup> parties will increase the need for a dependable and robust platform as well as new verification methods. Not having control of what software is running on the onboard telematics system can introduce great risks if not dealt with properly.

To identify the challenges with such open platform and the solution proposal for the problems, another set of interviews were conducted. During these 5 interviews, a whole new set of employees from the same Volvo companies were chosen. Another goal with the second set of interviews was to identify what parts of the system can be outsourced to 3<sup>rd</sup> party developers.

The interviewees unilaterally agreed that the important and critical parts of the telematics system should be developed in-house. A robust and well specified platform should be provided by Volvo for 3<sup>rd</sup> party developers. Furthermore there are requirements from the governments on some basic parts of the telematics systems that better not be outsourced to 3<sup>rd</sup> party developers. Some other parts can be developed either by 3<sup>rd</sup> party or in-house and are less crucial.

When introducing externally developed software, it is of great importance that the 3<sup>rd</sup> party components do not use all the system resources or deliberately or accidentally harm the functionality of more critical parts. The hardware resources in onboard telematics systems are limited and this makes it even more important to have robust and reliable resource management software.

According to most of the interviewees, an important goal for the vehicle companies should be to agree on standard interfaces for their telematics systems. This way the 3<sup>rd</sup> party components are usable on all vehicles which will increase

the efficiency for the vehicle companies, 3<sup>rd</sup> party companies and the end users.

The need for certification of 3<sup>rd</sup> party components was another solution proposal that was mentioned by the majority of the interviewees. Certification will ensure the robustness and reliability of the component to a certain level. Since the instability in these components can be experienced as instability in the Volvo system it is important to ensure the quality of the components. We will discuss certification briefly later in the paper.

##### B. Variation in deployment

Another key challenge and trend is to improve the dynamic update and installation of services during runtime in automotive telematics systems. Using this feature the end-users should be able to download or buy 3<sup>rd</sup> party software and install it on the onboard platform. Furthermore there should be possibility to update existing software.

In order to improve this option there are several prerequisites. Firstly the platform should be stable and continue to be stable after the installation. This requires a great deal of attention to resource management in a way that the newly installed component should not be able to harm the rest of the system. The system should even be functioning while downloading and installing the new component.

Secondly this puts some restrictions on how the component should be started and downloaded. Although the underlying platform should be robust this might not be sufficient in many cases due to the limited hardware resources. Careful resource management is needed during the download, install and start process.

#### V. TOWARDS SOLUTIONS

In this chapter we will discuss the existing solutions that can help facing the challenges mentioned earlier.

##### A. Reviews

Reviews and inspections are generally considered as very cost-effective verification methods and in recent years proposals have been made to make it even more so [15, 16]. Reviews are often most effective when focused on a specific viewpoint or aspect. We see a potential that checklists could be developed specifically for robustness and dependability reviews. This would be especially useful since it could help move robustness verification activities earlier in the development process. Even though some work has been done on this a general framework is needed which allows for company-specific adaptations [17].

##### B. Certification

Certification is mostly used to show the level of confidence in a system or component. Most of the certification models focus on mathematical and test based models. Using test cases the reliability and robustness of different components are analyzed to measure a reliability index for the system or component [18].



Using certification different characteristics of systems can be graded [18, 19]. One of these characteristics is robustness. This makes certification a good solution to consider when moving towards more open systems.

#### C. *Alignment of Requirements and Verification*

To ensure that the verification activities are aligned with the requirements previous research has focused mostly on ensuring traceability. However, there is a lack of results that are useful for non-functional requirements such as dependability and robustness. ‘Dependability Cases’ was developed in part to address these concerns [20]. However, they only indicate areas of importance and are not directly used to link the requirements to verification activities. By extending the non-functional use case idea and developing patterns for how to specify and test them much would be gained. Results on formal specification of safety patterns are related and can be built on [21].

#### D. *Formal methods*

Formal methods such as model checking and model-based specification have a potential to help address automotive telematics software challenges [22]. However, even though industrial use is increasing there is still a lack of methods that can help for whole-system and end-to-end verification.

#### E. *Automated Testing Tools for Robustness Testing*

The goal of robustness testing is to activate the design and programming or vulnerabilities in the system that result in incorrect operation. The robustness failures can be classified according to the CRASH scale [23, 24]: Catastrophic (the system crashes or reboots), Restart (the process hangs and needs to be restarted), Abort (the process aborts), Silent (No error signal returned when it should), and Hindering (incorrect error code is returned).

Robustness can be tested either by testing of exceptional input against the interfaces of the system or by stress testing the system with a large amount of valid input [23-26]. Robustness testing is typically either done as a part of the development process or it is performed after release to benchmark the robustness and dependability of the system or compare it to other existing systems [27].

Another use of robustness testing occurs when the system contains or plans to contain third party application or commercial off the shelf (COTS). Robustness testing in this case can help identifying what parts need wrapping. In many of these cases wrapping is the only solution to obtain higher robustness [25].

One of the earliest robustness testing methods was fault injection. The work started from very low level. In the early days most of the robustness testing tools tried to simulate hardware faults. More advanced versions of these tools are still used especially for testing embedded systems [14, 28].

Another low level and simple method based on generating random data as input for the system. Fuzz [29] was one of the pioneers in this field. Using Fuzz many robustness problems in Unix and Windows NT were discovered. This method became known as Fuzz testing.

Fuzz testing provides random input data to a program in order to test its robustness [30]. It is also called robustness testing or negative testing. It has been used to evaluate dependability and as an effective way of finding security defects. Recently, advances in symbolic execution and dynamic test data generation has been combined to create effective white box fuzz testing tools [30].

Another approach for robustness testing is providing the interfaces with invalid and out of bound values. Riddle tool [13] is one among many tools using this method. These tools usually use a grammar to define the range of valid and invalid inputs. The results showed a large number of unhandled exceptions of memory access violation and illegal instruction in large operative systems and their applications.

One of the most known tools for robustness testing is called Ballista [11]. Ballista and other testing tools in the same family are designed to perform more relevant automatic tests. In this method the valid and invalid inputs for the different parameters and types present in the system’s interfaces are specified. The hypothesis and result was that using these values the automatic tests performed are much better and more efficient in calculating and improving the robustness of the software system.

JCrasher [12], Check ‘n’ crash and DSD-crasher are other tools that more or less use the same principle as Ballista. However these are especially designed for the object oriented nature of Java. Using JCrasher different types and classes will automatically be instantiated to create an object oriented random input for the interfaces in the system.

Using these and other existing tools can help significantly towards a solution for robustness testing in automotive telematics. However developing and finding a powerful end-to-end testing tool that can initiate the tests from the onboard telematics system and follow the process back to the central servers at Volvo still remain an unsolved challenge.

## VI. CONCLUSIONS AND FUTURE WORK

Open telematics systems are rapidly growing in complexity and demand. Large infrastructure projects have been started by among others the European commission to address and guide this development. This has put demands and challenged the vehicle companies. In this paper we identified some of these trends and challenges that will have an impact on the robustness of the telematics systems. Furthermore we looked at the state of art and practice of robustness verification and discussed what demands these new trends would put on the robustness verification process.

A clear trend in automotive telematics industry is moving towards open systems that allow 3<sup>rd</sup> party developers to build components and new services for the system. These components should be available for the end user for purchase and installation. The introduction of these components brings new challenges for resource management and robustness of these systems.

There are several existing solutions and standards on such open platforms i.e. OSGI and CVIS. However, the robustness in these platforms has not been considered enough.

To face these challenges some existing and proposed solutions were discussed in this paper. An important step is to develop a robust platform. In order to achieve this goal there should be clear NFRs and specially robustness requirements that are measurable and verifiable. Furthermore, more sophisticated automated testing tools should be developed. Automated testing should be used in more extent to decrease the cost and time of verification and increase its quality. However, for the tools to be real-world attractive they should be useable for, or at least as parts of, end-to-end system testing. Another solution that has proven to increase the quality of software is to use reviews while developing the telematics platform.

Another step is to consider carefully what risks the 3<sup>rd</sup> party components can bring to the system. Using certification and testing the external components can be one way to face these risks. However, business concerns are important and will have to be considered. Working towards standard platforms and interfaces in the global automotive telematics industry can simplify this matter significantly.

In conclusion, moving towards more open telematics systems is inevitable in the automotive industry. In order to become successful, there is a need for more robust platforms and in order to verify this robustness, better testing and verification methods and tools are needed.

#### REFERENCES

- [1] J. C. Laprie, A. Avizienis, and H. Kopetz, "Dependability: Basic Concepts and Terminology," 1992.
- [2] A. Avizienis, J. C. Laprie, and B. Randell, "Fundamental concepts of dependability," 2001.
- [3] "Ieee standard glossary of software engineering terminology Tech. Rep. Std. 610.12-1990," 1990.
- [4] D. Sastry, G. Marco, L. Xuan, M. Paul, P. Ronald, S. Moninder, and T. Jung-Mu, "Framework for security and privacy in automotive telematics," in Proceedings of the 2nd international workshop on Mobile commerce Atlanta, Georgia, USA: ACM, 2002.
- [5] K. Grimm, "Software technology in an automotive company: major challenges," in Proceedings of the 25th International Conference on Software Engineering Portland, Oregon: IEEE Computer Society, 2003.
- [6] M. Broy, "Challenges in automotive software engineering," in Proceedings of the 28th international conference on Software engineering Shanghai, China: ACM, 2006.
- [7] Z. Yilin, "Telematics: safe and fun driving," Intelligent Systems, IEEE, vol. 17, pp. 10-14, 2002.
- [8] A. Gbr, "Autosar: Technical Review R3.1," 2008.
- [9] CVIS, "CVIS.1.0 Project Presentation," 2004.
- [10] O. Alliance, "Home page for OSGi Alliance and the OSGi Service Platform," 2009.
- [11] J. DeVale, P. Koopman, and D. Guttendorf, "The Ballista software robustness testing service," in Testing Computer Software Conference, 1999.
- [12] C. Csallner and Y. Smaragdakis, "JCrasher: An automatic robustness tester for Java," Software: Practice and Experience, 2004.
- [13] A. K. Ghosh and M. Schmid, "An approach to testing COTS software for robustness to operating system exceptions and errors," in Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on, 1999, pp. 166-174.
- [14] J. H. Barton, E. W. Czeck, Z. Z. Segall, and D. P. Siewiorek, "Fault injection experiments using FIAT," Computers, IEEE Transactions on, vol. 39, pp. 575-582, 1990.
- [15] A. Porter, H. Siy, C. A. Toman, and L. G. Votta, "An experiment to assess the cost-benefits of code inspections in large scale software development," in 3rd ACM SIGSOFT Symposium on Foundations of Software Engineering, Washington, D.C., United States, 1995.
- [16] E. Farchi and S. Ur, "Selective Homeworkless Reviews," in Software Testing, Verification, and Validation, 2008 1st International Conference on, 2008, pp. 404-413.
- [17] J. d. Almeida, J. B. Camargo, and B. Abrantes, "Best practices in code inspection for safety-critical software," IEEE Software, 2003.
- [18] A. Alvaro, E. S. de Almeida, and S. R. de Lemos Meira, "Software component certification: a survey," in Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on, 2005, pp. 106-113.
- [19] C. Wohlin and B. Regnell, "Reliability certification of software components," in Software Reuse, 1998. Proceedings. Fifth International Conference on, 1998, pp. 56-65.
- [20] R. A. Maxion and R. T. Olszewski, "Improving software robustness with dependability cases," in Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on, 1998, pp. 346-355.
- [21] F. Bitsch, Safety Patterns - The Key to Formal Specification of Safety Requirements vol. 2187/2001: Springer Berlin / Heidelberg, 2008.
- [22] L. M. Barroca and J. A. McDermid, "Formal Methods: Use and Relevance for the Development of Safety-Critical Systems," The Computer Journal, vol. 35, pp. 579-599, December 1, 1992 1992.
- [23] P. Koopman and J. DeVale, "The exception handling effectiveness of POSIX operating systems," Software Engineering, IEEE Transactions on, vol. 26, pp. 837-848, 2000.
- [24] P. Koopman, J. Sung, C. Dingman, D. Siewiorek, and T. Marz, "Comparing operating systems using robustness benchmarks," in Reliable Distributed Systems, 1997. Proceedings., The Sixteenth Symposium on, 1997, pp. 72-79.
- [25] N. P. Kropp, P. J. Koopman, and D. P. Siewiorek, "Automated robustness testing of off-the-shelf software components," in Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on, 1998, pp. 230-239.
- [26] ReSIST, "ReSIST NoE. Resilience-Building Technologies: State of Knowledge, Deliverable D12," 2006.
- [27] D. P. Siewiorek, J. J. Hudak, B. H. Suh, and Z. Segal, "Development of a benchmark to measure system robustness," in Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., The Twenty-Third International Symposium on, 1993, pp. 88-97.
- [28] T. K. Tsai, R. K. Iyer, and D. Jewitt, "An approach towards benchmarking of fault-tolerant commercial systems," in Fault Tolerant Computing, 1996., Proceedings of Annual Symposium on, 1996, pp. 314-323.
- [29] B. P. Miller, D. Koski, C. Pheow, and L. V. Maganty, "Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services," 1995.
- [30] Godefroid, "Automated Whitebox Fuzz Testing," 2008.

# A 2D-Barcode Based Mobile Advertising Solution

Jerry Zeyu Gao, Hema Veeraragavathatham, Shailashree Savanur, and Jinchun Xia  
Computer Engineering Department, San Jose State University  
Contact email: [jerrygao@email.sjsu.edu](mailto:jerrygao@email.sjsu.edu)

## Abstract

With the rapid increasing use of mobile devices, mobile-commerce has evolved at a very fast rate and greatly impacted our everyday lives. Since digital barcodes have been used extensively in traditional commerce, such as supply-chain and payment, it is desirable that we can continue to use the barcode technology in mobile commerce in advertising, payment, and product validation. This paper proposes a new mobile advertising solution based on two-dimensional (2D) barcodes. We report the design and implementation of our solution using DataMatrix 2D barcode technology. With this solution, merchants and manufactures, as well as advertising vendors can post and transfer barcode-based advertisements on mobile devices of end users.

**Keywords:** mobile advertising, mobile advertising system, mobile commerce, wireless commerce, and 2D barcodes.

## 1. Introduction

With the wide deployment of modern wireless networks and mobile technologies, the number of mobile device users increases sharply, which creates a strong demand for emerging mobile commerce applications and services. According to the November Issue of *Wireless Design & Development Asia* in 2007, wireless networks are deployed in 224 countries in the world. Informa Telecoms & Media (<http://www.marketresearch.com/vendors/>) reveals that worldwide mobile subscriptions will hit 3.3 billion—equivalent to 50 percent of the global. Mobile advertising is an important subject in mobile-commerce. It has received intense attention in today's advertising and commerce world. Informa Telecoms & Media predicts that wireless advertising revenue will grow from \$871 million in 2006 to over \$11 billion annually by 2011. Mobile devices have become the new frontier and hot targets for advertisers.

As indicated in [17], various barcodes have been used in the past decades as a very effective means in many traditional e-commerce systems, supply-chain management, retail sale-and-buy, as well as tracking and monitoring of products and goods. The barcode technology has evolved to 2-Dimensional that provides higher data capacity.

This paper reports our pioneer work of integrating the 2D barcode technology into mobile advertising systems. We extended our SmartMobile-AD [13] system by adding the 2D barcode technology to support encoding & decoding, posting & managing barcode-based mobile advertisements for advertisers, mobile service carriers and publishers.

The Barcode Based Mobile Advertising System proposed in this paper is a solution which could play a vital role in promoting finished goods over the mobile platform by

using the 2D barcode as the primary source for advertisements. This system facilitates advertisers to design their product advertisement (ad) and supply it to a mobile ad publisher to generate and publish a 2D barcode ad. The mobile customers can view and decode these 2D-barcode ads on mobile phones. They can also obtain 2d barcode ads from magazines or posters by using a camera-enabled mobile phone. If the ad is a promotion, then they can redeem it at the store by scanning the barcode using a barcode reader at the checkout counter.

This paper is structured as follows. Section 2 reviews the background of digital barcode technologies and discusses related work in mobile advertising. Section 3 describes our Barcode Based Mobile Advertising System, including its architecture, functionalities, and technologies used. The implemented workflow processes for advertisers, publishers, and vendors are illustrated in section 4. Section 5 presents several example applications of the implemented system for system users, including mobile access clients and online access clients. Finally, Section 6 concludes the paper and outlines future research directions.

## 2. Background and Related Work

### 2.1. Mobile advertising and systems

Mobile advertising provides a new direct way to increase product sales and the awareness of products and services by communicating with prospective buyers through mobile devices. The basic mobile advertising concepts were discussed in [1][2], including the classification of mobile ads, business models, challenges and opportunities. Various communication methods for mobile advertising were discussed in [4], such as *broadcasting (local and global)*, *ad hoc networking*, and *dedicated connection*. In addition, peer-to-peer based personalized advertisements can be sent to users using both push and pull modes.

Among all the papers published recently addressing mobile advertising, some focused on the effectiveness of mobile advertising. In [3], the authors drew the lessons learnt from analyzing the effectiveness of traditional advertising, in order to understand the same in mobile advertising. The authors considered a number of variables in order to develop an empirically validated model to study advertising through mobile messaging. The comparison between mobile advertising on mobile devices and conventional advertising on PCs and other platforms were discussed in [1]. Pousttchi and Wiedemann discussed the categorization and objectives of mobile marketing by examining fifty-five case studies [10]. They standardized mobile marketing into four types: information standard type (characterized by the instance information, for example, news and horoscopes), entertainment standard type (characterized by the instance

entertainment, for example, music and games), raffle standard type (characterized by the instance raffle, for example, lottery tickets), and coupon standard type (characterized by the instance monetary incentive, for example, discounts and trial packs).

Some other papers investigated the consumer's perspective and behavior toward mobile advertising and their acceptance [5][6][7]. In [9], the authors analyzed consumer behavior and the impact of a preceding advertisement on the current one. Using an applied research methodology they carried out an empirical evaluation of the model based on the data collected from the consumers. Another study was reported [8] to understand the response rates of consumers in Korea for the same service provider based on click rates of mobile advertisements, interviews, and literature reviews for mobile advertisements. In [15], the author provided a deep insight into the advertising space and their studies on the effectiveness of advertising campaigns for mobile phones. The author proposed a conceptual model based on the factors affecting the mobile industry and discussed the applicability of the model.

Different from the work summarized above, some other researchers devoted to practical solutions for mobile advertising, which is also the focus of this paper. The existing works can be classified into the following groups:

- **Mobile coupon system** [11], which issues and utilizes mobile coupons (known as mCoupons) on mobile devices. The system reported in [11] used Near Field Communication (NFC) technology for the client to extract the mCoupon from the issuer and interact with the target machine at the cashier's desk to cash-in the coupon.
- **Location-based advertising system.** Bluetooth-Mobile Advertising system [12], known as B-MAD, which delivers permission-based, location-aware mobile advertisements using Bluetooth positioning and Wireless Application Protocol (WAP) Push. In this system, the location of the end-user is identified through a Bluetooth Sensor. The Ad Server looks up any undelivered advertisements for that location, and then delivers the advertisements in the form of WAP Push messages. The authors in [14] analyzed the key issues in developing location-based advertisement for mobile e-commerce (Ad-me).
- **Intelligent mobile advertising system** [13], which is a complete system for both mobile marketing and advertising. It provides intelligent targeting rules and workflows to support various types of users including wireless service companies, ad publishers, and end-users.

## 2.2. Barcode technology for mobile advertising

The barcodes technology was invented decades ago. Traditionally, the barcodes stored data in the form of parallel lines in different widths, which are known as 1-dimensional (1D) barcodes and could only encode numbers [18]. In the past decade, various barcodes have been used as a very effective means in many traditional e-commerce systems, supply-chain management, retail sale-and-buy, as well as tracking and monitoring of products and goods. This technology has evolved into 2D barcodes [18] that can store large amount of data in a small area to support

information distribution and detection without accessing a database. However, 2D barcodes require sophisticated devices for decoding, which remained a challenge until recently. Today, with the advance of the image processing and multimedia capabilities of mobile devices, they can be used as portable barcode encoding and decoding devices.

With a much larger data capacity, 2D barcodes were quickly adopted in different areas. PDF417, Micro PDF417, and DataMatrix are typical examples of 2D barcodes. In general, there are two types of 2D barcodes: a) stacked 2D barcodes, such as Code 49 and PDF417, and b) Matrix 2D barcodes, such as Data Matrix and QR Code. Some examples of common 2D barcodes are shown below.

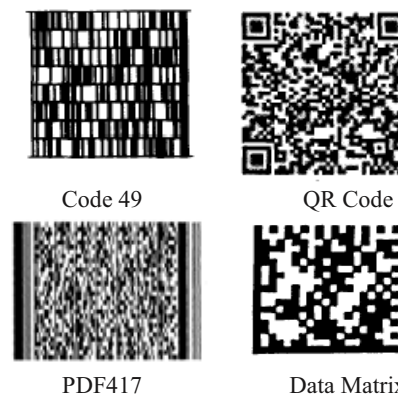


Figure 1 Four different Types of 2D Barcodes

A 2D barcode can hold up to 3116 digitals, 2355 letters, and 1556 binary data. As indicated in [17], many people believe digital barcodes improve mobile commerce application systems for the following reasons:

- Digital barcodes provide a simple and inexpensive method to present diverse commerce data in mobile commerce, including product id and product information, ads, and purchase/payment information.
- As more mobile digital cameras are deployed on mobile devices, using digital barcodes can reduce mobile inputs from mobile users.



Figure 2 Interactions between a 2D barcode and a Mobile phone (Source: [www.quickmark.com.tw](http://www.quickmark.com.tw))

As shown in Figure 2, the authors in [17] reviewed different mobile applications using 2D barcodes, in supply chain, mobile security, product identification and information tracking. Funk [16] forecasted the future of mobile shopping in the Japanese market as technology trajectories with regards to speed of the network. They include 2D barcodes, mobile browsers, and the integration

of mobile sites with other media, such as magazines, radio, and TV in the Japanese market.

### 3. 2D Barcode-Based Mobile Advertising

#### (A) Barcode-Based Mobile Advertising Process

Comparing with conventional advertising approaches, barcode-based advertising in mobile commerce provides **three** distinct advantages:

- Provide present diverse advertisements in small-sized barcodes with built-in rich marketing and product information for potential customers.
- Increase the convenience and efficiency of electronic commerce transactions in purchasing, payment, and delivery/pick-up for products with bar-coded IDs.
- Improve mobile user experience by reducing user inputs.

Figure 3 shows the business process and workflow for barcode-based mobile advertising. There are two ways for advertisers to post barcode-based advertisements. In **the first approach**, 2D barcode ads are published in conventional media (such as posters and magazines) by publishers. End users discover and capture the 2D barcode ads using mobile devices with digital camera. The mobile client software decodes the 2D barcodes and displays the content to end-users. After that, end users follow-up for e-commerce transactions.

In **the second approach**, 2D barcode ads are generated and posted by mobile ad publishers on mobile sites. When end-users discover a 2D barcode ad, they can click and access the advertisement contents decoded by mobile client software. Meanwhile, all end users reactions and responses are tracked and processed by the mobile advertising system for advertisement performance and reporting.

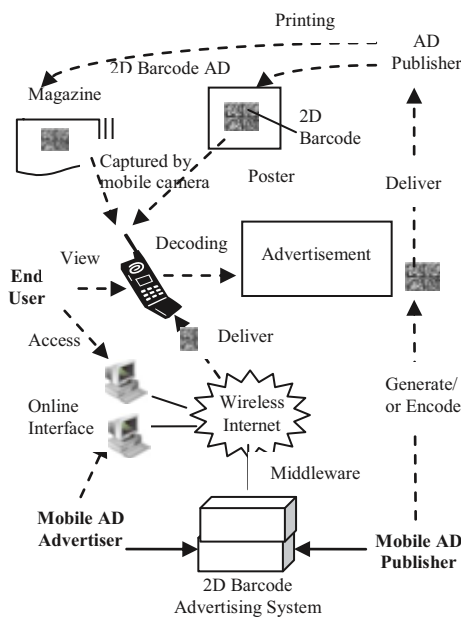


Figure 3. Barcode-Based Mobile Advertising Workflow

#### (B) Advertising Workflow

According to [2], there are three types of workflows to support advertising activities. They are: a) the enterprise-oriented workflow, b) the service-oriented workflow, and c) publisher-oriented workflow. The workflow process for mobile publishers consists six phases [1]:

- **Ad Space Catalog:** The publisher creates and maintains ad space catalogs. An ad space catalog consists of a number of pages with ad spaces. Each ad space has attributes including location, posting size, schedule, payment method, and current booking status.
- **Ad Space Trading:** The publisher sells ad spaces to advertisers following a set of business rules. Advertisers can select trading models through User Interfaces.
- **Ad Space Schedule:** The publisher creates and updates delivery schedules for each ad space based on the agreements. There are User Interfaces for advertisers to search, book, purchase and confirm ad schedules.
- **Ad Space Fulfillment:** The publisher delivers the ads based on ad delivery schedule and targeting rules.
- **Ad Space Measurement:** The publisher monitors and collects ad delivery data used to measure the performance of the ads.
- **Ad Space Payment:** The publisher collects payment from advertisers after the ad is delivered, according to the ad delivery contract.

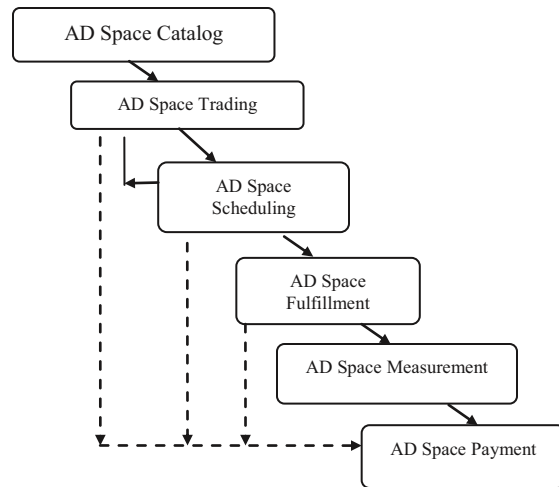


Figure 4 A Workflow Process for AD Publishers

#### Barcode-Based Technology Solution:

We started a project in 2006 to develop a 2D barcode mobile processing solution for our barcode-based mobile commerce system. From our literature survey and study about 2D barcodes, Data matrix symbology is a very popular technology and standard used in many applications. Hence, we implemented a 2D barcode framework based on the standard defined by the Information Technology-International Symbology Specification-Data Matrix (ISO/IEC 16022) [20]. The discussion of the algorithms was presented in [20]. The framework includes three basic components and one user interface as a barcode generation tool:

- An Application Interface (API), which can be used as a reusable component for 2D barcode processing.
- An encoding module, which includes two parts: *data encoder* and b) *image generator*. *Data encoder* performs data encoding functions. They are: a) generating encoded data in bit stream format, b) adding unprotected bit stream, c) calculating a header and a trailer, and appending them to the unprotected bit stream to produce protected bit stream, d) pattern randomization for extended security, and e) module placement to form the matrix using a special placement algorithm. *Image generator* includes the functions to create the 2D barcode image.
- A decoding module, which provides the decoding process by the following two parts: a) *image decoder*, and b) *data decoder*. *Image decoder* recovers the binary matrix from the given 2D barcode image.

Figure 5 shows the implemented encoding procedure and decoding procedure, where multi-string encoding and decoding feature were implemented to support multiple segment Data Matrix Barcodes.

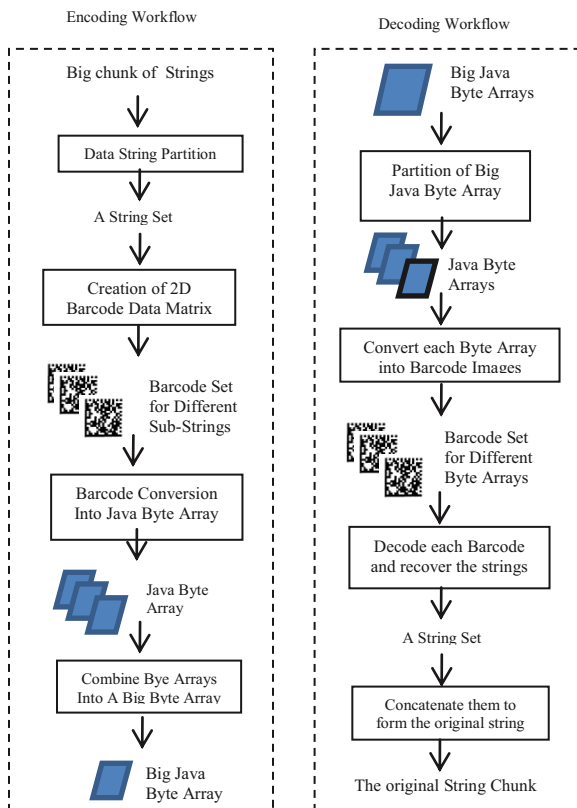


Figure 5 Encoding and Decoding for 2D Barcodes

#### 4. A Barcode-Based Mobile AD Solution

Since 2005, we have started the research project [13] to create a wireless advertising system to support wireless service companies and publishers (or portals) to accept, deliver, and present mobile ads for mobile advertisers. In 2008, we added the following two major features:

- Support, manage, post, and deliver 2D barcode-based mobile ads.
- Support, manage, and post location-based ads. Due to the page limit, this paper only discusses the 2D barcode-based advertising solutions.

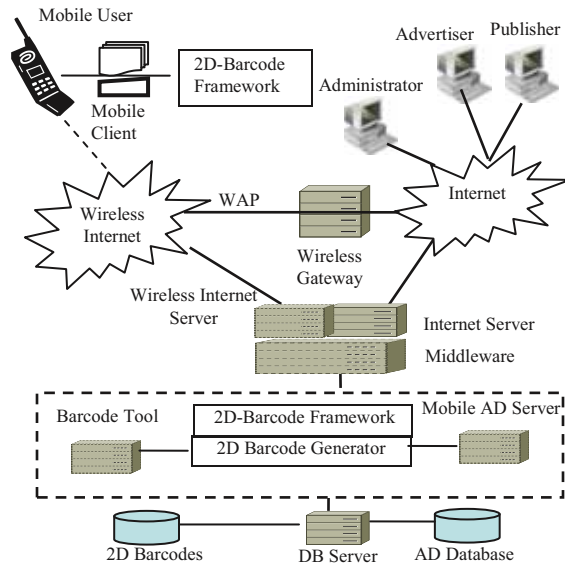


Figure 6 The Mobile Advertising System Architecture

Figure 6 demonstrates the system infrastructure, in which WAP-based wireless Internet is used to support the communications between mobile clients and the mobile advertising server. The system supports three types of users: mobile advertisers, ad publishers, and mobile users. As shown in Figure 4, the system supports both online and mobile functionalities. The function components are grouped in 3-tier layers.

Tier #1 – A client tier including online user interface and mobile client interface software.

Tier #2 – The application tier, which contains a set of functional components and necessary middleware, including wireless internet server, and internet server, etc.

These functional components support both online and mobile advertising functions. Details are shown in Figure 7. Tier #3 – A data store tier, which includes a mobile AD database program and a MySQL server.

As shown in Figure 7, the functional components in the mobile advertising server can be classified into two groups: (a) online functional features, and (b) mobile functional features. The online features include the following parts:

Ad Space Catalog Manager – It creates, updates, and maintains ad spaces for a mobile publisher.

Ad Schedule Manager – It creates, updates, and maintains different advertising schedules for mobile advertisers, including 2D barcode advertisements.

Ad Submission Manager – It allows advertisers to submit mobile ads based on their selected ad templates, contents and spaces. The system allows advertisers to preview any mobile ads on a mobile emulator from a computer.

Ad Targeting Manager – This component is responsible to select mobile ads based on the pre-defined targeting rules and processes, which were explained in [13].

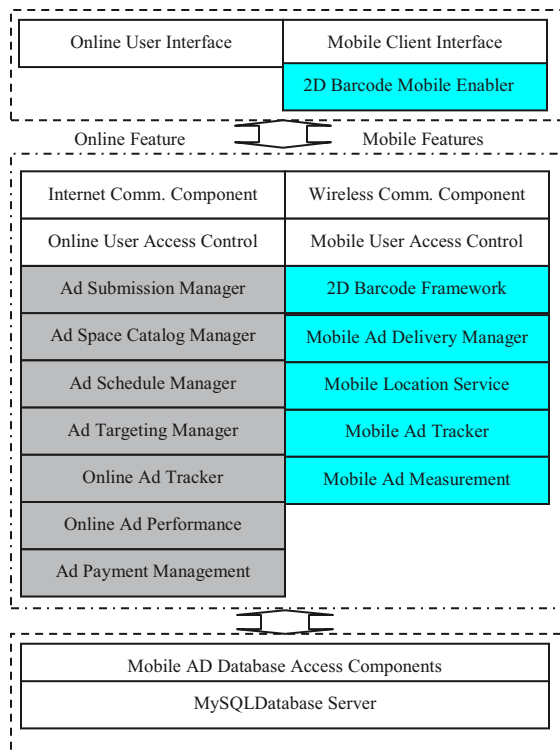


Figure 7 Function Components in 3-Tier Layers

Online Ad Tracker – This part allows advertisers to track the posted ads and customer reactions online.

Online Ad Performance –It evaluates and measures posted ad performance based on different criteria (such as, ad-impression, ad-click and ad-click-through) and generates different performance reports.

Compared to the previous version of this Mobile AD system [13], five mobile components were added. They are listed as follows.

- *2D Barcode Framework* – This component provides 2D barcode related functions and facilities using Data Matrix standard, including encoding and decoding functions. It is used as a basic component on the sever side of the mobile advertising system.
- *2D Barcode Mobile Enabler* – This is a barcode processing component for mobile client software, which supports decoding, barcode-based information retrieval, and interactions with mobile users.
- *Mobile Ad Delivery Manager* – This component manages and delivers different types of barcode-based mobile advertisements.
- *Mobile Ad Tracker* – This function component provides a primitive tracking for mobile advertisements. It helps advertisers to check the current advertising process and posting status for a selected mobile ad.
- *Mobile ad Measurement* – This function component allows advertisers to access a simple ad performance report for their posted ads.

## 5. Implementation and Application

The 2D barcode advertising system demonstrates the use of the barcode ads in various forms namely, large banner, small banner, icon, text, image and video. Some of the screen layouts are represented in Figure 8. The application is launched with a splash screen and a list displaying the various categories namely, Clothing and Apparel, Household Items, Personal Care, Electronics, Pet Care, and Beverages. Each category has two or three advertisement links. These links navigate to forms that display ads on index or home pages of different mobile sites. The user should click on the View Ad button in order to view the decoded barcode. The user might have to view videos before viewing the decoded barcode ad.

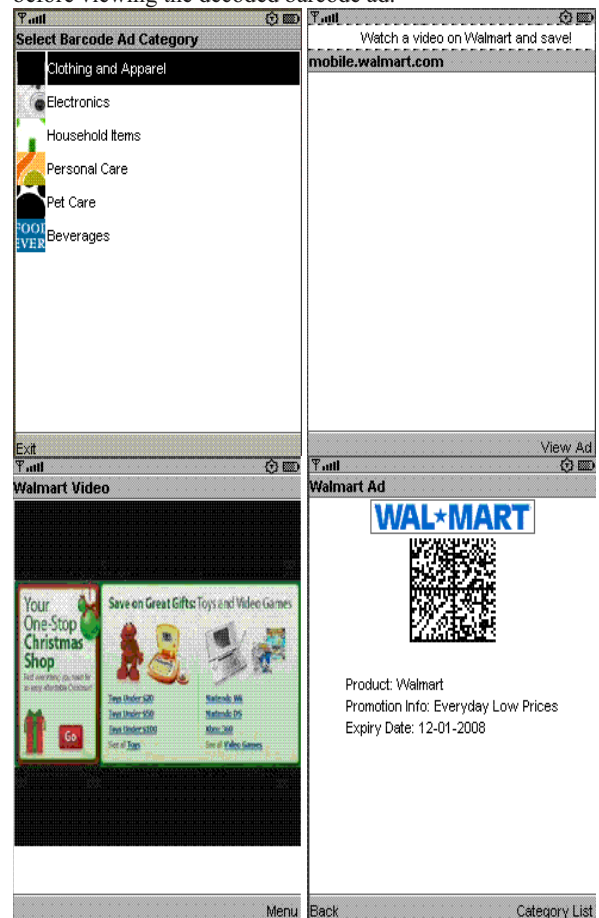


Figure 8. Screen layouts for customer to view various barcode ads

The system provides a means for the advertisers to plan and generate advertisements that belong to specific categories and subcategories. The advertiser can also choose registered publishers, ad templates, ad spaces and ad schedules as posted by the publisher on his or her ad catalogs. The advertiser changes the status of the barcode ad to “contract” as soon as he completes the design. The publisher receives the barcode ad and then encodes it. Figure 9 illustrates the online user interface for a mobile ad advertiser to manage mobile ad categories. Figure 10 shows the system screen layout to generate a contract for mobile

advertising. Figure 11 and 12 demonstrate the online interface for ad publishers to list/view mobile ads, and track mobile ads, respectively. In addition, a mobile ad performance feature is implemented online to allow users to check the performance of mobile advertisements using an online interface.



Figure 9 A Screen Layout for Managing AD Categories



Figure 10 A Screen Layout to Generate A Contract

The successful implementation of the barcode based advertising system illustrates the fact that 2D barcode based advertisements can be delivered to devices with very small memory footprints. Encoding the advertisement into a barcode ensures that more ad content can be delivered to the mobile devices. Because the customers choose the categories in which they wish to receive barcode ads, we can avoid ad spam and ensure that the advertisers generate more targeted and informational ads.

### Implementation Technology

The presented 2D barcode advertising solution was implemented using J2EE (Java 2 Enterprise Edition) (for the web client) and J2ME (Java 2 Micro Edition) (for the mobile client). Netbeans IDE was used to develop the

entire application. JSP pages were developed for the web client and CSS was used to provide an uniform and elegant look. Sun Java wireless toolkit for Connected Limited Device Configuration (CLDC) was used to develop the mobile application. Java Servlet and EJB technology were used to develop the business logic. JDBC was used to connect to a MySQL database.



Figure 11 A Screen Layout for Publishers to List and View Mobile Barcode Ads



Figure 12. A Screen Layout to Track Information of Mobile Barcode Ads

The encoding algorithm used by the Publisher was developed earlier at San Jose State University. The implementation of decoding for the Data Matrix barcode is currently based on an open source API, namely the ZXing Barcode Reader for Java ME which is a part of Google code. It is recommended that an in-house barcode reader be used in order to own the entire solution.

The solution currently implements only text-based advertisements. It is recommended that multimedia based ads be supported in order to reach a wider range of customers.



## 6. Conclusion and Future Work

Today, mobile advertising is a hot topic in m-commerce research and business. As the advance of 2D barcode technologies, more barcode-based solutions and systems are needed in mobile commerce. This paper addresses this demand and reports our research efforts in building a 2D barcode-based mobile advertising solution for mobile advertisers and mobile users using modern mobile technology based on DataMatrix 2D barcode standard. Compared with other existing mobile advertising approaches and solutions, the proposed solution has the unique advantages in increasing mobile AD access experience for the following reasons:

- Posting small-size 2D barcode-enabled ads on mobile devices allows mobile users to access and retrieve a rich set of information, including detailed product information in a supply-chain.
- Improving mobile commerce experience by reducing or eliminating mobile user inputs when accessing mobile ads.
- Creating a new digital channel to leverage the conventional advertising media (such as posters and magazines) with mobile advertising solutions.

This paper reports our development effort and experience in building a new mobile advertising solution based on 2D barcodes. Initial prototype system and application experience suggest that this solution is feasible over wireless Internet using current technologies in 2D barcode symbology and mobile J2ME. Our experiment results also suggest that the 2D barcode technology has great potential in mobile commerce systems, such as mobile payment and mobile applications.

In the future, we plan to study, implement, and deploy the 2D barcode-based mobile validation solution and tool to support merchants in product validation & check-out, product pick-up & delivery, invoice (or ticket) validation by mobile devices.

## 7. References

- [1] Yunos, H. M., J. Z. Gao, and S. Shim, "Wireless Advertising's Challenges and Opportunities", *IEEE Computer*, 36 (5), 30-37, 2003.
- [2] Jerry Gao, J., Shim S., Mei H., Su X., "Engineering Wireless-Based Software Systems and Applications", Artech House Publishers, 2006.
- [3] D. Drossos and M. G. Giaglis, "Mobile Advertising Effectiveness: an Exploratory Study", *Proceedings of the International Conference on Mobile Business*, 2006.
- [4] R. Bulander, M. Decker, G. Scheifer, and B. Kolmel, "Comparison of Different Approaches for Mobile Advertising", *Proceedings of the second IEEE International Workshop on Mobile Commerce and Services*, 2005.
- [5] X. Shen, and H. Chen, "An Empirical Study of What Drives Consumers to Use Mobile Advertising in China" *Proceedings of the third International Conference on Grid and Pervasive Computing Workshops*.
- [6] D. He and Y. Lu, "Consumers Perceptions and Acceptances Towards Mobile Advertising: An Empirical Study in China", *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing*, 2007.
- [7] H. Komulainen, et al, "Mobile advertising in the eyes of retailers and consumers - empirical evidence from a real-life experiment", *Proceedings of the International Conference on Mobile Business*, 37-37, 2006.
- [8] P. Hagharian et al., "Increasing Advertising Value of Mobile Marketing – An Empirical Study of Antecedents", *Proceedings of the thirty-eighth Hawaii International Conference on System Sciences*, 32(c) – 32(c), 2005.
- [9] S. H. Lee., et al, "Analysis of the Actual Response Rates in Mobile Advertising", *Innovations in Information Technology*, 1-5, 2006.
- [10] K. Pousttchi and D. G. Wiedemann, "Contribution to Theory Building for Mobile Marketing: Categorizing Mobile Marketing Campaigns through Case Study", *Proceedings of the International Conference on Mobile Business*, 2006.
- [11] S. Dominkus, and M. Aigner, "mCoupons: An Application for Near Field Communication (NFC)", *Proceedings of the twenty-first International Conference on Advanced Information Networking and Applications Workshops*, 2007.
- [12] L. Aalto et al, "Bluetooth and WAP push based location-aware mobile advertising system", *Proceedings of the International Conference on Mobile systems, applications and services*, 49-58, 2004.
- [13] Jerry Zeyu Gao and Angela Ji, "SmartMobile-AD: An Intelligent Mobile Advertising System", *Proceedings of the third International Conference on Grid and Pervasive Computing Workshops*, 2008.
- [14] N. Hristova and G. M. P. O'Hare, "Ad-me: wireless advertising adapted to the user location, device and emotions", *Proceedings of the thirty-seventh Annual Hawaii International Conference on System Sciences*, 2004.
- [15] R. Vatanparast, "Piercing the Fog of Mobile Advertising", *Proceedings of the International Conference on the Management of Mobile Business*, 2007.
- [16] L. J. Funk, "The future of mobile shopping: The interaction between lead users and technological trajectories in the Japanese market", *Technological Forecasting and Social Change*, 74 (3), 341-356., 2007.
- [17] Jerry Zeyu Gao, L. Prakash, and R. Jagatesan, "Understanding 2D-BarCode Technology and Applications in M-Commerce - Design and Implementation of A 2D Barcode Processing Solution", *Proceedings of the thirty first International Conference on Computer Software and Applications*, 2007 (COMPSAC2007).
- [18] R. C. Palmer, *The Bar Code book: Reading, Printing, and Specification of Bar Code Symbols* (3rd ed.), Helmers Publishing, 1995.
- [19] H. Kato, and K. T. Tan, "2D BARCODES FOR MOBILE PHONES", *Proceedings of the second International Conference on Mobile Technology, Applications and Systems*, 2005.
- [20] ISO/IEC, "INTERNATIONAL STANDARD ISO/IEC 16022 – DataMatrix", June 2006.

# Long-term Prediction of Wireless Network Traffic

Zhiwei Xu  
University of Michigan  
Dearborn, MI 48187  
zwxu@umich.edu

Zhou Zhou  
Department of Statistics  
The University of Chicago  
Chicago, IL 60637 USA  
zhouzhou@uchicago.edu

Weibiao Wu  
Department of Statistics  
The University of Chicago  
Chicago, IL 60637 USA  
wbwu@galton.uchicago.edu

## Abstract

We consider the problem of predicting aggregates or sums of future values of a wireless network based on its past values. In contrast with the conventional prediction problem in which one predicts a future value given past values of the process, in our setting the number of aggregates can go to infinity with respect to the number of available observations. Consistency and Bahadur representations of the prediction estimators are established. A simulation study is carried out to assess the performance of different prediction estimators.

## 1 Introduction

Efficient prediction of network traffic is an important problem for planning and for Quality-of-service (QoS) improvement. Net work traffic can be considered as a time series. So prediction of future traffic can be modeled as the prediction or forecasting of future values of a random process. This is one of the fundamental objectives in the study of time series. Let  $(X_t)_{t \in \mathbb{Z}}$  be a stochastic process. Given the observations  $X_1, \dots, X_n$ , one is interested in predicting future values  $X_{n+j}$ ,  $j \geq 1$ . If  $(X_t)_{t \in \mathbb{Z}}$  is stationary with  $\mathbb{E}(X_t^2) < \infty$ , then one can apply the celebrated Kolmogorov-Wiener theory to estimate the conditional mean  $\mathbb{E}(X_{n+j}|X_1, \dots, X_n)$ . Since Cover (1975), there have been substantial progresses on the estimation theory of the conditional mean  $\mathbb{E}(X_{n+j}|X_1, \dots, X_n)$ , the conditional distribution  $[X_{n+j}|X_1, \dots, X_n]$ , or their variants; see for example Ornstein (1978), Ryabko (1988), Algoet (1992, 1999), Morvai, Yakowitz and Györfi (1996), Morvai, Yakowitz and Algoet (1997), Györfi, Morvai and Yakowitz (1998), Györfi, Lugosi and Morvai (1999), Schäfer (2002), Morvai (2003), Morvai and Weiss (2004, 2005, 2008). Morvai and Weiss (2004, 2005) applied the tool of stopping times to estimate conditional expectations. Other contributions can be found in Brockwell and Davis (1991), Box,

Jenkins and Reinsel (1994), Györfi et al (1998), Pourahmadi (2001) and Györfi and Ottucsak (2007) among others.

In this paper, we consider predicting  $X_{n+1} + \dots + X_{n+m}$ , sum of future values, based on the past observations  $X_1, \dots, X_n$ . In our setup we allow  $m = m_n \rightarrow \infty$  as  $n \rightarrow \infty$ . Our formulation is attractive in situations in which one is interested in long-term prediction. For example, in telecommunication, engineers may have automatically collected minutely or secondly time series data which represent number of downloads by users every minute or second. However, at the management level, people are more interested in predicting numbers of downloads for a much longer time scale, say weekly, monthly or even yearly. Let  $X_1, X_2, \dots, X_n$  be minutely observations and  $m = 7 \times 24 \times 60 = 10,080$ . Then  $X_{n+1} + \dots + X_{n+m}$  corresponds to the number of downloads in the upcoming week. Prediction of  $X_{n+1} + \dots + X_{n+m}$  may help implementing price policy of telecommunication services. Recently, the *Time Warner Cable Inc* is planning to implement a price policy which is based on Internet usage and download volumes rather than a flat monthly fee (Adegoke, 2008). To design a reasonable price policy, one needs to have a good prediction of  $X_{n+1} + \dots + X_{n+m}$ .

As a mathematical framework, we consider the construction of prediction intervals for  $X_{n+1} + \dots + X_{n+m}$  given  $X_1, \dots, X_n$ . Specifically, we need to find a random interval  $[L, U]$ , where  $L$  and  $U$  are functions of  $X_1, \dots, X_n$ , such that

$$\mathbb{P}(L \leq X_{n+1} + \dots + X_{n+m} \leq U | X_1, \dots, X_n) = 1 - \alpha, \quad (1)$$

where  $1 - \alpha$  is a pre-assigned coverage level. In practice one typically uses  $\alpha = 0.01$  or  $0.05$ . Clearly, the problem of finding such  $L$  and  $U$  involves the estimation of conditional distributions of  $X_{n+1} + \dots + X_{n+m}$  given  $X_1, \dots, X_n$ . With the estimated interval  $[L, U]$ , one can assess uncertainty of future aggregates and then adopt appropriate price policies.

In our setting we let  $m \rightarrow \infty$  as  $n \rightarrow \infty$ . Our framework is very different from the classical one in which one

assumes  $m = 1$  as far as the methods of finding  $L$  and  $U$  are concerned. If  $m = 1$ , then one needs to estimate the conditional distribution of  $X_{n+1}$  given  $X_1, \dots, X_n$ . The latter problem is closely related to the Value-at-Risk (VaR) estimation problem; see J.P. Morgan's (1996) RiskMetrics Technical Report. In a typical conditional VaR estimation problem, analogously to (1), one seeks to find a number  $V$  which depends on  $X_1, \dots, X_n$ , such that

$$\mathbb{P}(X_{n+1} > V | X_1, \dots, X_n) = \alpha. \quad (2)$$

It turns out that, interestingly, estimation of  $L$  and  $U$  in the framework of (1) with  $m \rightarrow \infty$  is relatively easier for certain class of processes. This is due to the so-called quenched or conditional central limit theory; see Wu and Woodroffe (2004) for some recent developments. As argued in Section 2.1, if the process  $(X_k)$  is weakly dependent, then the impact of  $X_1, \dots, X_n$  on the sum  $X_{n+1} + \dots + X_{n+m}$  is negligible and one has the approximate relation

$$\begin{aligned} \mathbb{P}(X_{n+1} + \dots + X_{n+m} \leq x | X_1, \dots, X_n) \\ \approx \mathbb{P}(X_{n+1} + \dots + X_{n+m} \leq x) \end{aligned} \quad (3)$$

when  $m$  is large. Let  $l < u$  be two real numbers such that

$$\mathbb{P}(l \leq X_{n+1} + \dots + X_{n+m} \leq u) = 1 - \alpha. \quad (4)$$

In many problems approximate solutions  $l$  and  $u$  can be obtained asymptotically or empirically; see Section 2. Based on (3), we can choose  $L$  and  $U$  as  $l$  and  $u$ , respectively, so that they provide an approximate solution to (1).

We now impose structural assumptions on  $X_i$  so that we can interpret in what sense (3) holds and then utilize (3). In particular, we shall consider the long-term prediction for the linear model

$$X_i = w_i^T \beta + e_i, \quad (5)$$

where  $T$  denotes the matrix transpose,  $(e_i)$  is a mean zero stationary process,  $\beta$  is a  $p \times 1$  unknown regression coefficient vector and  $w_i$  are known  $p \times 1$  covariates, explanatory variables or design vectors.

The rest of the paper is organized as follows. As a premier, Section 2 concerns the special case of model (5) in which  $w_i^T \beta = 0$ , namely there are no covariates involved. Prediction of the general linear model (5) is considered in Section 3. In Section 3.1 we apply our estimation procedure to a telecommunication network traffic dataset.

## 2 Quantiles of Sums of Stationary Processes

To illustrate the idea behind (3), we let  $w_i^T \beta = 0$  and assume that  $(e_i)$  is a mean zero stationary process with finite

second moment  $\mathbb{E}(e_i^2) < \infty$ . Let  $S_m = e_1 + \dots + e_m$  and  $\mathcal{F}_i = (e_i, e_{i-1}, \dots)$ . Under this setting, by stationarity, it suffices to establish the following version of (3):

$$\mathbb{P}(S_m/\sqrt{m} \leq x | \mathcal{F}_0) \approx \mathbb{P}(S_m/\sqrt{m} \leq x) \quad (6)$$

when  $m$  is large. Let  $\Phi(u) = \int_{-\infty}^u (2\pi)^{-1/2} e^{-x^2/2} dx$  be the standard normal distribution function. For a random variable  $X$ , we write  $X \in \mathcal{L}^p$ ,  $p > 0$ , if  $\|X\|_p := [\mathbb{E}(|X|^p)]^{1/p} < \infty$ . For two distributions  $F$  and  $G$  on  $\mathbb{R}$ , define the Levy metric

$$\begin{aligned} \Delta(F, G) = \inf\{\delta > 0 : F(x - \delta) - \delta \leq G(x) \\ \leq F(x + \delta) + \delta \text{ holds for all } x \in \mathbb{R}\}. \end{aligned} \quad (7)$$

### 2.1 Quenched Central Limit Theory

Wu and Woodroffe (2004) proved the following conditional or quenched central limit theorem: Assume that  $\mathbb{E}(|e_i|^p) < \infty$  for some  $p > 2$  and, for some  $q > 5/2$ ,

$$\|\mathbb{E}(S_m | \mathcal{F}_0)\|_2 = O(\sqrt{m}/\log^q m). \quad (8)$$

Then we have the almost sure convergence

$$\Delta[N(0, \sigma^2), \mathbb{P}(S_m/\sqrt{m} \leq \cdot | \mathcal{F}_0)] \rightarrow 0 \quad (9)$$

as  $m \rightarrow \infty$ , where  $\sigma^2 = \lim_{m \rightarrow \infty} \|S_m\|_2^2/m$  is the long-run variance. Namely, for almost all realizations of  $\mathcal{F}_0$ , we have

$$\lim_{n \rightarrow \infty} \sup_{x \in \mathbb{R}} |\mathbb{P}(S_m/\sqrt{m} \leq x | \mathcal{F}_0) - \Phi(x/\sigma)| = 0. \quad (10)$$

Convergence in the stronger form of invariance principle is also valid; see Corollary 3 in Wu and Woodroffe (2004). As argued in the latter paper, under (8), we also have the unconditional central limit theorem  $S_m/\sqrt{m} \Rightarrow N(0, \sigma^2)$ , or:

$$\lim_{n \rightarrow \infty} \sup_{x \in \mathbb{R}} |\mathbb{P}(S_m/\sqrt{m} \leq x) - \Phi(x/\sigma)| = 0. \quad (11)$$

Clearly, (10) and (11) imply that not only (6) holds in the sense of

$$\lim_{n \rightarrow \infty} \sup_{x \in \mathbb{R}} |\mathbb{P}(S_m/\sqrt{m} \leq x | \mathcal{F}_0) - \mathbb{P}(S_m/\sqrt{m} \leq x)| = 0. \quad (12)$$

but also both  $\mathbb{P}(S_m/\sqrt{m} \leq x | \mathcal{F}_0)$  and  $\mathbb{P}(S_m/\sqrt{m} \leq x)$  can be approximated by  $N(0, \sigma^2)$ . The latter observation is in striking contrast with the construction of prediction intervals when  $m = 1$ , in which case the conditional and unconditional versions are quite different (see Section 7.4 in Chatfield (2001)). Additionally, (10) and (11) also suggest an approximate solution of  $L$  and  $U$  to (1) in the following form:

$$L, U = \pm \hat{\sigma} z_{\alpha/2} \sqrt{m}, \quad (13)$$

where  $z_{\alpha/2}$  is the  $\alpha/2$ -th quantile of the standard normal distribution and  $\hat{\sigma}$  is an estimate of  $\sigma$ . A popular estimate of  $\sigma^2$  is the lag window estimate

$$\hat{\sigma}^2 = \sum_{k=-k_n}^{k_n} \hat{\gamma}_k, \quad (14)$$

where  $k_n$  is the bandwidth sequence satisfying  $k_n \rightarrow \infty$  and  $k_n/n \rightarrow 0$ , and  $\hat{\gamma}_k$  is an estimate of  $\gamma_k$ :

$$\hat{\gamma}_k = \frac{1}{n} \sum_{i=1}^{n-|k|} (e_i - \bar{e})(e_{i+|k|} - \bar{e}), \quad \bar{e} = \frac{1}{n} \sum_{i=1}^n e_i. \quad (15)$$

For details see Anderson (1971) or Brockwell and Davis (1991).

An interesting and useful feature of the prediction interval (13) is that one does not need to fit the underlying probability model for the process  $(e_i)$ . On the other hand, however, the above normal approximation may fail if the process is strongly dependent or has heavy-tailed distributions. It is common that telecommunication time series may exhibit long-range dependence as well as heavy tails; see for example Mikosch et al (2002). This is one of the major reasons that Internet Service Providers such as the *Time Warner Cable Inc* are interested in imposing more charges on users who download files with very large sizes. To construct prediction intervals for processes with heavy tails, one way out is to resort to empirically based method which is discussed in detail in the section below. Our simulation study shows that the latter approach outperforms the one based on (13).

## 2.2 Quantile Estimates

Condition (8) ensures the normal approximation (9). For strongly dependent processes, however, (8) is violated (Wu and Woodroffe, 2004) and (12) may be invalid. In this case, we propose to estimate quantiles of  $S_m$  by sample quantiles of  $\sum_{j=i-m+1}^i e_j$ ,  $i = m, m+1, \dots$ , via a moving window scheme. Specifically, Let

$$\tilde{Y}_i = \frac{\sum_{j=i-m+1}^i e_j}{H_m}, \quad i = m, m+1, \dots, \quad (16)$$

where  $H_m > 0$  is an appropriate normalizing constant such that  $\tilde{Y}_i$  has a non-degenerate limiting distribution as  $m \rightarrow \infty$ . Note that  $(\tilde{Y}_i)$ ,  $i \in \mathbb{Z}$ , is a (triangular array) stationary time series and we can calculate  $(\tilde{Y}_i)_m^n$ . In order to construct a  $(1 - \alpha)$  prediction interval for  $\tilde{Y}_{n+m}$  based on  $(\tilde{Y}_i)_m^n$ , we shall estimate  $(1 - \alpha/2)th$  and  $(\alpha/2)th$  quantiles of this quantity. More specifically, let  $\hat{Q}_n(\alpha/2)$  and  $\hat{Q}_n(1 - \alpha/2)$  be the  $(1 - \alpha/2)th$  and  $(\alpha/2)th$  sample quantiles of  $(\tilde{Y}_i)_m^n$ , then  $[\hat{Q}_n(\alpha/2), \hat{Q}_n(1 - \alpha/2)]$  is a natural  $(1 - \alpha)$  prediction interval of  $\tilde{Y}_{n+m}$ . Therefore

$[H_m \hat{Q}_n(\alpha/2), H_m \hat{Q}_n(1 - \alpha/2)]$  is a  $(1 - \alpha)$  prediction interval for  $\sum_{j=n+1}^{n+m} e_j$ . Asymptotic properties of  $\hat{Q}_n$  for short- and long-range dependent linear processes are dealt with in Sections 2.2.1 and 2.2.2, respectively.

### 2.2.1 Short-Range Dependent (SRD) Processes

To obtain asymptotic properties of  $\hat{Q}_n(\alpha/2)$  and  $\hat{Q}_n(1 - \alpha/2)$ , here we assume that  $(e_i)$  is a one-sided infinite order moving average MA( $\infty$ ) process:

$$e_i = \sum_{j=0}^{\infty} a_j \varepsilon_{i-j}, \quad (17)$$

where  $(\varepsilon_j)_{j=0}^{\infty}$  is an i.i.d. sequence having mean 0, and  $(a_i)_{i=0}^{\infty}$  are real coefficients such that  $e_i$  exists almost surely. The almost sure convergence of (17) can be checked by the well-known Kolmogorov three-series theorem (Chow and Teicher (1988)).

The innovations  $(\varepsilon_j)_{j=0}^{\infty}$  can be either light or heavy-tailed. More precisely, we say  $\varepsilon_j$  is light-tailed if  $\mathbb{E}(\varepsilon_j^2) < \infty$ . For heavy-tailed processes, we consider  $\varepsilon_j$  which belongs to  $\alpha$ -stable domain of attraction  $\mathcal{D}(\alpha)$  for some  $\alpha \in (1, 2)$ , namely the normalized partial sum process of  $\varepsilon_j$  converges to a stable distribution (Chow and Teicher (1988), Feller (1971)). For  $\varepsilon_j \in \mathcal{D}(\alpha)$ , it has the following characterization:

$$\begin{aligned} 1 - F_\varepsilon(t) &= (c_1 + o(1))t^{-\alpha}L(t) \text{ and} \\ F_\varepsilon(-t) &= (c_2 + o(1))t^{-\alpha}L(t) \end{aligned} \quad (18)$$

as  $t \rightarrow \infty$ , where  $F_\varepsilon(\cdot)$  is the cumulative distribution function (c.d.f.) of  $\varepsilon_j$ ,  $c_1, c_2 \geq 0$ ,  $c_1 + c_2 > 0$  and  $L$  is a slowly varying function (s.v.f.), i.e.,  $\lim_{x \rightarrow \infty} L(tx)/L(x) = 1$  for all  $t > 0$ ; see Feller (1971). Clearly, by (18),

$$g_n := \inf\{x : \mathbb{P}(|\varepsilon_i| > x) \leq 1/n\} = n^{1/\alpha}L_1(n),$$

where  $L_1$  is also a s.v.f.. Observe that  $\mathbb{E}(|\varepsilon_i|^{\alpha'}) < \infty$  for all  $\alpha' \in (0, \alpha)$ , and  $\alpha$  is called the heavy tail index, and  $\mathbb{E}(\varepsilon_i^2) = \infty$ .

We shall let the normalizing constant  $H_m = \sqrt{m}$  if  $\mathbb{E}(\varepsilon_j^2) < \infty$  and  $H_m = g_m$  if  $\varepsilon_j \in \mathcal{D}(\alpha)$ ,  $1 < \alpha < 2$ . By the central limit theorem of SRD linear processes (see for example Avram and Taquq (1984)), we have

$$\tilde{Y}_i \Rightarrow Z \text{ as } m \rightarrow \infty, \quad (19)$$

where  $Z$  is Gaussian if  $\mathbb{E}(\varepsilon_j^2) < \infty$  and  $Z$  is  $\alpha$ -stable if  $\varepsilon_j \in \mathcal{D}(\alpha)$ .

We shall impose the following regularity conditions:

(SRD)  $\sum_{i=0}^{\infty} |a_i| < \infty$ .

(DEN)  $\sup_{x \in \mathbb{R}} (f_\varepsilon(x) + |f'_\varepsilon(x)|) < \infty$ , where  $f_\varepsilon(\cdot)$  is the density of  $\varepsilon_i$ .

Condition (SRD) is a classic short-range dependence or stability condition for linear processes (see Box, Jenkins and Reinsel (1994)). The other Condition (DEN) appears quite mild, and it holds, for example, the symmetric- $\alpha$ -stable distributions. Let  $\tilde{Q}_q$ ,  $0 < q < 1$ , be the  $q$ th quantile of  $\tilde{Y}_i$  and  $f_m(\cdot)$  be the density function of  $\tilde{Y}$ . Let

$$\tilde{F}_n(x) = \frac{1}{n-m+1} \sum_{i=m}^n I\{\tilde{Y}_i \leq x\} \quad (20)$$

be the empirical distribution function of  $(\tilde{Y}_i)_m^n$ . We have the following two theorems regarding the asymptotic behavior of  $\hat{Q}_n$  in the light and heavy-tailed cases, respectively.

### 2.2.2 Long Range Dependent (LRD) Processes

It is common in the literature to call process (17) long memory or long range dependent if the coefficients  $(a_i)_0^\infty$  are not absolutely summable or in other words, if  $\sum_{i=0}^\infty |a_i| = \infty$ . In this section we shall consider the following decay of the series  $(a_i)_0^\infty$ :

- (LRD)  $a_i = i^{-\gamma} l(i)$ ,  $i = 1, 2, \dots$ ,  $\lambda < \gamma < 1$ , where  $l(\cdot)$  is a s.v.f. and  $\lambda = 1/2$  if  $\mathbb{E}(\varepsilon_i^2) < \infty$  and  $\lambda = 1/\alpha$  if  $\varepsilon_i \in \mathcal{D}(\alpha)$ ,  $1 < \alpha < 2$ .

That  $\lambda < \gamma$  is necessary for the almost sure convergence of (17) and the other constraint  $\gamma < 1$  is to guarantee that  $(a_i)_0^\infty$  is not absolutely summable and hence long memory of the series  $(e_i)$ . An important class of models which satisfy condition (LRD) is the the fractionally integrated ARIMA (FARIMA) processes (Hosking (1981)).

In the long memory case, define the normalizing constants  $H_m = m^{3/2-\gamma} l(m)$  if  $\mathbb{E}(\varepsilon_i^2) < \infty$  and  $H_m = g_m m^{1-\gamma} l(m)$  if  $\varepsilon_i \in \mathcal{D}(\alpha)$  for some  $\alpha \in (1, 2)$ . Then central limit results of  $(e_i)$  holds under the above normalization in the sense of (19). See Taqqu (2003) and Avram and Taqqu (1984). We have the following theorems in the LRD case. The latter theorems are different from the ones in the SRD case in that  $\gamma$ , the parameter controlling the strength of dependence, is needed in the condition of  $m$ .

## 3 Prediction of Linear Models

Consider now model (5). We shall predict  $X_{n+1} + \dots + X_{n+m}$  based on  $X_1, X_2, \dots, X_n$ . The latter observations can be used to estimate the unknown parameter vector  $\beta$ . Specifically, let  $W = (w_1, \dots, w_n)^T$  be the design matrix. Then the least squares estimate (LSE) of  $\beta$  has the form

$$\hat{\beta}_{ls} = (W^T W)^{-1} W^T X. \quad (21)$$

When the errors  $(e_i)$  are heavy-tailed, it is more desirable to use robust estimates of the regression coefficient

$\beta$  (see Huber (1981)). Therefore in this situation we suggest using the least absolute distance (LAD) estimation of  $\beta$ ; namely let

$$\hat{\beta}_{lad} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n |X_i - w_i^T \beta|. \quad (22)$$

The LAD estimation is equivalent to the median regression for which fast and stable algorithms are available; see Koenker (2005).

In both the LSE and LAD cases, the estimated residuals can be written as

$$\hat{e}_i = X_i - w_i^T \hat{\beta}, \quad i = 1, 2, \dots, n. \quad (23)$$

It is hoped that the procedures proposed in Sections 2.1 and 2.2 can be applied to the residuals and hence the prediction interval for  $X_{n+1} + \dots + X_{n+m}$  can be obtained. More precisely, we propose the following procedure:

- (i) Use the LAD procedure to obtain  $\hat{\beta}$  and  $\hat{e}_i$ ,  $i = 1, 2, \dots, n$ .
- (ii) Let  $\tilde{Y}_i = \sum_{j=i-m+1}^i \hat{e}_j$ ,  $i = m, \dots, n$ . Obtain the  $\alpha/2$  and  $1 - \alpha/2$  empirical quantiles of  $(\tilde{Y}_i)_m^n$ , denoted by  $\tilde{Q}_n(\alpha/2)$  and  $\tilde{Q}_n(1 - \alpha/2)$ .
- (iii) A  $(1 - \alpha)$  prediction interval for  $\sum_{i=n+1}^{n+m} X_i$  can be constructed as  $\sum_{i=n+1}^{n+m} w_i^T \hat{\beta} + [\tilde{Q}_n(\alpha/2), \tilde{Q}_n(1 - \alpha/2)]$ .

Note the LSE can be used in step (i) when the errors are light tailed in the hope of gaining more efficiency. Residual quantile-quantile (QQ) plots can be used to check the tail of the errors  $(e_i)$ , while more sophisticated methods can also be adopted for the purpose. Since in the regression case we have to estimate the errors  $(e_i)$  by the residuals  $\hat{e}_i$ , we need to investigate whether the consistency property of the empirical quantiles listed in Theorems ??-?? still hold in this case.

Let  $\tilde{Y}_i = \sum_{j=i-m+1}^i \hat{e}_j / H_m$ ,  $i = m, \dots, n$ , where  $H_m$  is defined as in Section 2.2. Define  $\tilde{Q}_n(q)$  as the  $q$ th empirical quantile of  $(\tilde{Y}_i)_m^n$ . We have the following theorem regarding the asymptotic behavior of the  $\tilde{Q}_n(q)$ 's.

**Theorem 1.** Let  $\Sigma_n = W^T W$ . Assume that (a) there exists constants  $0 < C_s < C_l < \infty$ , such that  $C_s < \lambda_1 \leq \lambda_n < C_l$  for all large  $n$ , where  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$  are the eigenvalues of  $\Sigma_n/n$ ; that (b) there exists a constant  $C^* < \infty$ , such that  $\max_{1 \leq i \leq n} |w_i| \leq C^*$  for all large  $n$ ; that (c)  $f_e(0) > 0$ , where  $f_e(\cdot)$  is the density function of  $e_i$ . Then conclusion (i) in Theorems 1 to 4 holds with  $\hat{Q}_n(q)$  therein replaced by  $\tilde{Q}_n(q)$ .

Treat the heavy tail index  $\alpha = 2$  in the light tail case and the long memory index  $\gamma = 1$  in the SRD case. From the proof of Theorem 1, we see that under the conditions of the

latter theorem  $\hat{\beta}_{lad} - \beta = O_p(n^v)$  for all  $v > 1/\alpha - \gamma$ . Hence  $\sum_{i=n+1}^{n+m} w_i^T (\hat{\beta} - \beta) = O_p(mn^v) = o_p(1)$  under conditions of Theorems ??-??. On the other hand, by Theorem 1 and the discussions in Remark ??,  $H_m \hat{Q}_n(q) - H_m \tilde{Q}_q = o_p(1)$ . Therefore we conclude that the lower bound  $\sum_{i=n+1}^{n+m} w_i^T \hat{\beta} + \tilde{Q}_n(\alpha/2)$  in (iii) is a weakly consistent estimator of the  $\alpha/2$ th quantile of  $\sum_{i=n+1}^{n+m} X_i$ . Analogous conclusion holds for the upper bound.

**Remark 1.** An interesting and useful feature of our construction procedure (i)-(iii) of the long-run prediction interval is that we do not need to estimate the tail index and long memory index. Estimation of the latter indices are important and highly nontrivial. The simplicity and generality of the quantile estimation method further justify its use in practice.  $\diamond$

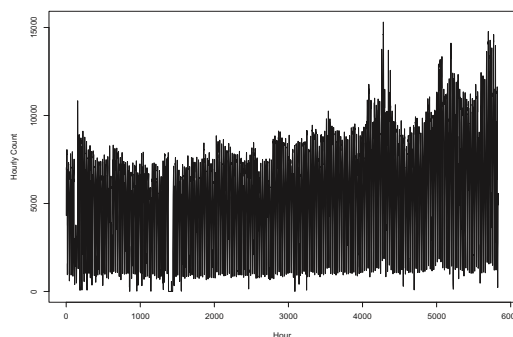
**Remark 2.** Conditions (a) and (b) in Theorem 1 on covariates are general enough for many applications. For example, they are satisfied under the design  $w_i = (1, (i/n)^a, \cos(i/b), \sin(i/b))^T$  for  $a > -1$  and  $b > 0$ , which are the covariates chosen for the Motorola telecommunication network traffic dataset discussed in Section 3.1. On the other hand, by changing condition (b) to  $\max_{1 \leq i \leq n} |w_i| \leq C^* \log n$  for all large  $n$ , we allow random design  $(w_i)$  with exponentially decaying tails. In this case conclusions of Theorem 1 continue to hold as long as we add an extra factor of  $\log n$  into the probability bounds of  $\hat{Q}_n(q) - \tilde{Q}_q$  there.  $\diamond$

### 3.1 Prediction of Wireless Network Traffics

In this section we shall apply the quantile estimation procedure to the Motorola telecommunication network traffic dataset from a mobile infrastructure network deployed in Asia and US. The network is designed for mobile users to conduct voice communication and to download digital items (ring tones, wall paper, music, video, games, etc) to their mobile devices. As multimedia cell phones and pocket PCs become popular, there is an increasing demand for digital items. However, mobile users cannot download digital items directly from the third party content provider (TPCP) network. They need to go through wireless access point provided by their wireless service provider to access TPCP websites. The wireless networks that handle both voice and digital items are more complex, expensive and they require more bandwidth than traditional networks that handle voice only. Knowing the traffic trend is critical to the management and long-term prediction will be useful for resource allocation, maintenance plan and price policy.

We collected the traffic data of the eight months period from 11:00AM July, 2005 to 10:59AM March 8 2006, and we obtained 5832 hourly transaction counts (see Figure 1 for a plot of the hourly counts). Our purpose is to construct

a 95% prediction interval for the total usage of the week following the 8-month period. This amounts to predicting the sum of  $m = 24 \times 7 = 168$  future values. Figure 1 gives the time series plot of the hourly counts.



**Figure 1. Time series plot of the hourly counts.**

It is noticeable from Figure 1 there is an abnormal period around hour 1400 (around September 5th 2005) with consecutive low counts. We checked the system log and found out it was due to system outage. Other outliers can be caused by system maintenance, outage, upgrade, etc. Our LAD regression procedure and the quantile estimation method are resistant to the occurrence of outliers (See Koenker (2005)).

The hourly data exhibits strong periodicity of 24 as the wireless usage peaks at about 8pm and minimizes at about 5am every day. There is also an noticeable increasing pattern of the usage. We choose the following regression model

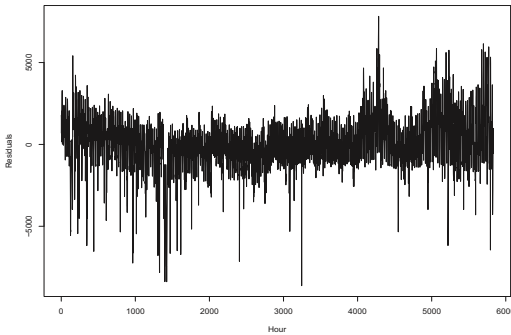
$$X_i = \beta_1 + \beta_2 \sqrt{i/5832} + \beta_3 \cos(2\pi i/24) + \beta_4 \sin(2\pi i/24) + e_i, \quad i = 1, 2, \dots, 5832. \quad (24)$$

The estimated coefficients  $\hat{\beta}_1 = 3429$ ,  $\hat{\beta}_2 = 2811.9$ ,  $\hat{\beta}_3 = 3498.7$  and  $\hat{\beta}_4 = -771.1$ . Figure 2 below shows the residuals of model (24).

Following steps (i)-(iii) in Section 3, we obtain a 95% prediction interval for the usage of the following week as [771714.7, 1297852]. Note again that the latter interval is not the 95% confidence interval for the mean of usage of the following week.

## 4 Conclusion

We propose quantile regression to perform robust prediction of network traffic data. The quantile regression procedure has several promising features: it is resistant to outliers; it reveals distributional information of the predicted



**Figure 2. Residual plot of model (24).**

values and it gives an accurate long-term coverage probabilities. These features are of essential importance in planning and QoS improvements.

## REFERENCES

- Adegoke, Y (2008) Time Warner to test Internet billing based on usage. *Reuters*, Jan 16, 2008.
- Algoet, P. (1992) Universal schemes for prediction, gambling and portfolio selection. *Ann. Probab.* 20, 901–941.
- Algoet, P. (1999) Universal schemes for learning the best nonlinear predictor given the infinite past and side information. *IEEE Trans. Inform. Theory* 45, 1165–1185.
- Anderson, T. W. (1971). *The Statistical Analysis of Time Series*. John Wiley, New York.
- Avram, F. and Taqqu, M. S. (1986). Weak convergence of moving averages with infinite variance, in: Eberlein and Taqqu (Eds.), *Dependence in Probability and Statistics: A Survey of Recent Results*. Birkhauser, Boston, pp. 399–416.
- Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1994) *Time series analysis. Forecasting and control*. Third edition. Prentice Hall, Inc., Englewood Cliffs, NJ.
- Breidt, F. J.; Davis, R. A. and Dunsmuir, W. (1995) Improved bootstrap prediction intervals for autoregressions. *J. Time Ser. Anal.* 16 177–200.
- Brockwell, P.J. and Davis, R.A. (1991). *Time Series: Theory and Methods, second edition*. Springer-Verlag, New York.
- Chatfield, C. (2000) *Time-Series Forecasting*. Chapman and Hall/CRC, Boca Raton, FL.
- Chow, Y. S. and Teicher, H. (1988). *Probability Theory*. Springer Verlag, New York.
- Cover, T. M. (1975) Open problems in information theory, In: *1975 IEEE Joint Workshop on Information Theory*, IEEE Press, New York, 1975, pp. 3536.
- Dalrymple, D. J. (1987) Sales forecasting practices: Results from a United States survey Pages. *Int. J. Forecasting* 3 379–391.
- Fan, J. and Q. Yao (2003) *Nonlinear Time Series*, Springer, New York.
- Feller, W. (1971). *An introduction to probability theory and its applications. Vol. II*. John Wiley & Sons, New York.
- Freedman, D. A. (1975). On tail probabilities for martingales. *Annals of Probability* 3 100–118.
- Gnedenko, B. V. (1954). Gnedenko, B. V. A local limit theorem for densities. (Russian) *Doklady Akad. Nauk SSSR* 95, 5–7.
- Györfi, L., Härdle, W., Sarda, P. and Vieu, P. (1989) Non-parametric curve estimation from time series. *Lecture Notes in Statistics*, 60. Springer-Verlag, Berlin, 1989.
- Györfi, L., G. Lugosi and G. Morvai, (1999) A simple randomized algorithm for consistent sequential prediction of ergodic time series, *IEEE Transactions on Information Theory* 45, 2642–2650.
- Györfi, L., Morvai, G. and Yakowitz, S.J. (1998). Limits to consistent on-line forecasting for ergodic time series. *IEEE Transactions on Information Theory*, 44, 886–892.
- Györfi, L. and Ottucsák, G. (2007). Sequential prediction of unbounded stationary time series. *IEEE Transactions on Information Theory*, Volume: 53, Issue: 5 1866–1872.
- Hahn, G. J. and W. Meeker (1991) *Statistical Intervals: A Guide for Practitioners*, Wiley, New York.
- Hannan, E.J. (1973) Central limit theorems for time series regression, *Z. Wahrsch. Verw. Gebiete* 26 157–170.
- Hosking, J. R. M. (1981). Fractional differencing. *Biometrika* 68 165–176.
- Huber, P. J. (1981) *Robust statistics*. Wiley, New York.
- Ibragimov, I. A. and Linnik, Yu. V. (1971). *Independent and stationary sequences of random variables*. Wolters-Noordhoff.
- Koenker, R. (2005). *Quantile regression*. Cambridge University Press, Cambridge.
- Mikosch, T., Resnick, S., Rootzén, H. and Stegeman, A. (2002), Is network traffic approximated by stable Levy motion or fractional Brownian motion? *Annals of Applied Probability* 12 23–68.
- Morgan, J. P. (1996). *Value at Risk*, RiskMetrics Technical Document, New York.
- Morvai, G. (2003) Guessing the output of a stationary binary time series, in: *Foundations of Statistical Inference*. Y. Haitovsky, H.R. Lerche, Y. Ritov (Eds.) (2003), Physika Verlag, 205–213.
- Morvai, G. and Weiss, B. (2004) Intermittent estimation of stationary time series. *Test* 13, 525–542.

- Morvai, G. and B. Weiss (2005) Inferring the conditional mean. *Theory Stoch. Process.* 11, 112–120
- Morvai, G. and Weiss, B. (2008) On universal estimates for binary renewal processes. *Ann. Appl. Probab.* 18, 1970–1992.
- Morvai, G., S. Yakowitz, and P. Algoet (1997) Weakly convergent nonparametric forecasting of stationary time series, *IEEE Transactions on Information Theory* 43, 483–498.
- Ornstein, D. (1978) Guessing the next output of a stationary process. *Israel J. Math.* 30, 292–296.
- Pourahmadi, M. (2001). *Foundations of Time Series Analysis and Prediction Theory*. Wiley, New York.
- Ryabko, B. Ya. (1988) Prediction of random sequences and universal coding, *Problems of Inform. Trans.* 24, 3–14.
- Schäfer, D. (2002) Strongly consistent online forecasting of centered Gaussian processes, *IEEE Transactions on Information Theory* 48, 791–799.
- Schmoyer, R. L. (1992) Asymptotically valid prediction intervals for linear models. *Technometrics* 34 399–408.
- Taqqu, M. S. (2003). Fractional Brownian motion and long-range dependence, in: P. Doukhan, G. Oppenheim and M. S. Taqqu (Eds.), *Theory and Applications of Long-range Dependence*. Birkhauser, Boston.
- Wu, W. B. (2005). On the Bahadur representation of sample quantiles for stationary sequences. *The Annals of Statistics* 33 1934-1963.
- Wu, W. B. (2007). M-estimation of linear models with dependent errors. *The Annals of Statistics* 35 495-521.
- Wu, W. B., Michailidis, G. and Zhang, D. (2004). Simulating Sample Paths of Linear Fractional Stable Motion. *IEEE Transactions on Information Theory*, Volume: 50, Issue: 6 1086-1096.
- Wu, W. B. and Woodroffe, M. (2004) Martingale approximations for sums of stationary processes. *Annals of Probability* 32 1674-1690
- Vardeman, S. (1992) What about the other intervals? *The American Statistician* 46 193-197.
- Zagdański, A. (2005) On the construction and properties of bootstrap- $t$  prediction intervals for stationary time series. *Probab. Math. Statist.* 25 133–153.
- Zhou, Z. and Wu, W. B. (2007). On linear models with long memory and heavy-tailed errors. Preprint.
- Zhou, Z. and Wu, W. B. (2009). Local linear quantile estimation of non-stationary time series. *Annals of Statistics*, To Appear.



# Resource Allocation for a Modular Software System

Lance Fiondella and Swapna S. Gokhale  
Department of Computer Science & Engineering  
University of Connecticut  
Storrs, CT 06269  
{lfiondella,ssg}@engr.uconn.edu

## Abstract

*Most existing software optimization research assumes advance knowledge of the component parameters. Perfect future knowledge of fault detection is an unnecessary oversimplification and greatly diminishes the applicability of the previously proposed techniques. While efficient resource allocation is essential, these studies ignore the importance of components within a software's architecture. In this paper we present an adaptive optimization procedure that periodically assesses the testing process and allocates time to components. Using a case study, we illustrate how our approach adapts, increasing resources to a component when more failures are observed during testing. The results suggest that this interactive method is responsive to clusters of faults encountered during testing and can help to minimize the time needed to optimize software.*

## 1 Introduction

Predominant software engineering processes rely on component based development. To improve the reliability of an application, it is necessary to improve the reliability of the constituent components. The resulting allocation problem must divide available resources between components while considering several practical factors. These include the criticality of the component within the application architecture and the amount of time needed to realize component improvements.

Previous software reliability optimization studies [1, 2, 3] commonly assume advance knowledge of the parameters of a software reliability model. This removes the major source of real world uncertainty and it greatly diminishes the utility of the proposed techniques. During the software development and testing process, resource allocation decisions must be made based on the limited data available up to that point. Faults persist and new ones are discovered as testing progresses. Thus, resource allocation decisions must periodically adapt by committing time to components most in need of attention. Another shortcoming of previous approaches is

that many do not consider component interactions.

Our approach acknowledges both software architecture and uncertainty in the fault detection process, implementing an iterative test allocation procedure with multiple checkpoints. We propose an adaptive optimization procedure for modular software that periodically assesses and efficiently provisions testing time to components. Time allocation functions are derived from the conditional reliability expression of a non-homogeneous Poisson process model. The optimization procedure iteratively applies the Dale-Winterbottom (DW) [4] procedure. A detailed illustration demonstrates the approach. The results indicate that periodic allocation accounts for architecture in achieving a target reliability, while adapting to clusters of faults encountered during testing. The adaptive DW helps minimize time needed to achieve a target reliability when the fault detection processes are not known with certainty.

The layout of the paper is as follows: Section 2 provides an overview of our modeling approach. Reliability optimization is addressed in Section 3. A detailed illustration is given in Section 4. Related research is summarized in Section 5. Section 6 provides conclusions and future research.

## 2 Modeling approach

In this section, we present our approach to model system reliability in terms of component reliabilities, within the context of the system architecture. We then present the relationship between the reliability of a component and its testing time. We conclude the section with a definition of the time-adjusted reliability importance measure.

### 2.1 System-level modeling

We consider a terminating application which operates on demand. It is possible to distinguish between consecutive runs for such an application. We let  $n$  denote the number of components in the application, and assume that the architecture of the application is represented by the one-step transition probability matrix of a Discrete Time Markov Chain (DTMC) [5]. An analysis of the DTMC representing the

application architecture can provide the expected number of visits to each component during a typical execution of the application [5]. Let  $\nu_i$  and  $r_i$  denote the expected number of visits and reliability of component  $i$ . The expected system reliability  $E[R]$  is then given by [6]:

$$E[R] = \prod_{i=1}^n r_i^{\nu_i}. \quad (1)$$

## 2.2 Component-level modeling

Equation (1) suggests that estimates of component reliabilities are necessary to compute system reliability. The reliability of a component, in turn, is determined by the time-dependent failure behavior. We use the common Goel-Okumoto (GO) software reliability model [7] to estimate the reliability of a component based on its observed failure behavior as a function of testing time. For component  $i$ , the mean value function  $m_i(t_i)$  of the GO model, which provides the expected number of failures observed by testing time  $t_i$  is given by Equation (2). The parameters  $\alpha_i$  and  $\beta_i$  respectively denote the number of faults that will eventually be detected and the fault detection rate of component  $i$ .

$$m_i(t) = \alpha (1 - e^{-\beta t}) \quad (2)$$

Based on the mean value function, the reliability of component  $i$  Component reliability is given by Equation (3), where  $t_{mni}$  is the normalized mission time.

$$r_i = R(t_i + t_{mni} | t_i) = e^{-(m_i(t_i + t_{mni}) - m_i(t_i))} \quad (3)$$

The normalized mission time for a component  $i$  is computed by scaling the total mission time  $t_{mn}$ , using the expected number of visits to the component (obtained from analyzing the DTMC representing the application architecture) as follows:

$$t_{mni} = \frac{\nu_i}{\sum_{i=1}^n \nu_i} \times t_{mn}. \quad (4)$$

For a given mission time  $t_{mn}$ , Equation (3) can be used to compute the present reliability  $r_i$  based on the estimated model parameters  $\alpha_i$  and  $\beta_i$ . Increasing the component reliability requires additional testing. Solving Equation (3) the projected time  $t_i$  for which the component needs to be tested to achieve a given reliability  $r_i$  is given by Equation (5). Thus, Equation (3) directly models the relationship between the testing time of the component and its reliability.

$$t_i(\alpha_i, \beta_i, t_{mni}, r_i) = \frac{1}{\beta_i} \log \left( \frac{\alpha_i (1 - e^{-\beta_i t_{mni}})}{\log \left( \frac{1}{r_i} \right)} \right) \quad (5)$$

## 2.3 Time-Adjusted Reliability Importance

Equation (5) indicates that the extent to which testing time expended on component  $i$  improves its reliability is a function of its parameters  $\alpha_i$  and  $\beta_i$ . Furthermore, the extent to which improvement in the reliability of component  $i$  improves system reliability depends on system architecture, which is embodied in the system reliability expression of Equation (1). In an ideal scenario, large improvements in component reliability can be achieved by expending as little time as possible, while small improvements in component reliability will lead to large improvements in system reliability. To consider the complex impact of component testing times and reliabilities, and system architecture, we define the time-adjusted reliability importance measure [8] as follows:

$$\begin{aligned} d_i &= \frac{dt_i(\alpha_i, \beta_i, t_{mni}, r_i)}{dr_i} \left[ \frac{\partial E[R]}{\partial r_i} \right]^{-1} \\ &= \left( -r_i \beta_i \log(r_i) \log(\nu_i) \prod_{i=1}^n r_i^{\nu_i} \right)^{-1}. \end{aligned} \quad (6)$$

In Equation (6), the numerator is the rate of increase in time with respect to reliability for component  $i$ . This is modeled by Equation (5), which increases exponentially as  $r_i \rightarrow 1$ . This term in Equation (6) deters from allocating time to components that are either hard to improve or are already highly reliable and therefore would require substantial time to make them even more reliable. The denominator is the Birnbaum importance measure [9], which identifies component improvements that are most critical to enhancing system reliability. The system reliability expression of Equation (1) is used to compute the Birnbaum measure. It is important to note that improving the reliability of one component will change the Birnbaum importance of all components due to the complex interactions among the components. Preferably, the numerator will be small and denominator large. This situation would indicate that the amount of time needed for making big reliability improvements is not increasing very fast and that the realizable gains at the system level are large respectively. In other circumstances, the time needed to improve a component's reliability could be substantial, yet still be a component to target because it plays a central role in the proper operation of the application making its sensitivity very high.

In order to improve system reliability with minimal testing time, intuition suggests that the least amount of time should be spent on the components that are most responsive and will produce the greatest gains in system reliability. Furthermore, the components that are rarely used (have a very small Birnbaum importance), but are very easy to improve should be avoided. This corresponds to concentrating on the components with smallest  $d_i$  of Equation (6). The time-adjusted reliability importance thus provides an objective measure to select components to achieve the desired system

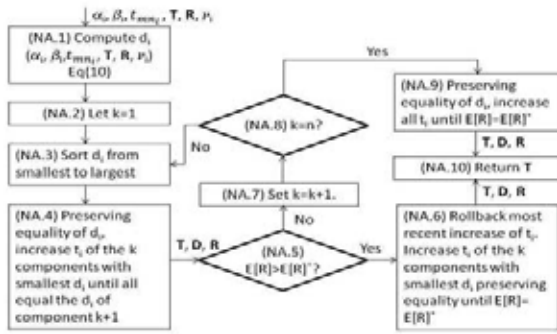


Figure 1. Dale-Winterbotton procedure

reliability target. Such an objective component selection approach is especially necessary in the later stages when all the components have a relatively high reliability and require a significant amount of time to improve. Thus, Equation (6) should help to identify the optimal  $\mathbf{T}^* = (t_1^*, t_2^*, \dots, t_n^*)$  such that a reliability objective is achieved while minimizing the time spent on improving components.

### 3 Reliability optimization

In this section, we first explain the Dale-Winterbotton (DW) [4] procedure to achieve the desired system-level reliability with minimal time. We then explain why the DW procedure is not adequate to optimize the reliability of software systems. We conclude the section with a presentation of an adaptive procedure to optimize the reliability of a software system, which embeds the DW procedure.

#### 3.1 Dale-Winterbotton Procedure

The DW procedure determines the testing time that needs to be allocated to each component of a system in order to achieve the desired reliability target. It only requires a simultaneous root finding procedure. For each component  $i$ , the procedure accepts as input the expected number of visits ( $\nu_i$ ), the normalized component mission times ( $t_{mn_i}$ ), the current testing time ( $t_i$ ), component parameter estimates ( $\alpha_i$  and  $\beta_i$ ), and associated reliability estimates ( $r_i$ ). A step-wise explanation of the DW procedure, which are also depicted in the flowchart in Figure 1, is as follows.

- **DW.1** Initialize  $k$  to 1.
- **DW.2** Compute  $d_i$  using Equation (6).
- **DW.3** Sort  $d_i$  in ascending order. Since small  $d_i$ s are preferable,  $d_1$  is identified as the component that can improve system reliability in the least amount of time.

- **DW.4** Formulate a system of  $k = 1$  equations, by setting  $d_1$  equal to  $d_2$ .  $d_1$  and  $d_2$  respectively denote the first and the second least time consuming components. Holding the reliabilities in the  $n - 1$  harder to improve components, the only unknown in Equation (6) for the least time consuming component is its reliability. Solving the equation produces a new higher value of reliability, and substituting it into Equation (5) provides the new (increased) value of testing time.
- **DW.5** Using Equation (1), check if this additional time used to improve the reliability of the least time consuming component overshoots the system reliability target  $E[R]^*$ .
- **DW.6** If the system reliability target overshoots, the projected time allocated to the least time consuming component is rolled back and the time of the easiest component is increased just enough to achieve the system reliability target. In this situation, we use the equations  $E[R] = E[R]^*$ , with reliabilities held at their original values for the remaining harder to improve  $n - 1$  components. The last projected allocation overshoot the reliability target, which implies that the target system reliability can be precisely satisfied by allocating less time. Equation (1) identifies the reliability of the easiest component needed to achieve the system target.
- **DW.7** If no overshoot occurred, increment  $k$  to 2 and consider the impact of improving the two most responsive components.
- **DW.8** If  $k < n$ , then continue from **DW.3**. The  $d_i$ s will be sorted again to identify a third least time consuming component  $d_3$  that will serve as a ceiling for worthwhile allocation of time to the two least time consuming components  $d_1$  and  $d_2$ . This process is repeated iteratively until  $k = n$  or reliability overshoot occurs when improving  $k < n$  components. For a general case when  $k < n$  components are to be improved simultaneously, the reliability of the  $n - k$  components with the highest  $d_i$  are held constant and a system of  $k$  equations is formulated. These equations are of the form  $d_i = d_{k+1}$ , where  $d_i \in \{1, 2, \dots, k\}$  are the adjusted reliability importance of the  $k$  easiest to improve components and  $d_{k+1}$  is the importance of the  $(k + 1)^{st}$  least time consuming component. When an overshoot occurs in the process of improving  $k < n$  components, the previous projected time allocation is rolled back and the time of the  $k$  easiest components are increased in a manner which satisfies the equations  $d_1 = d_i$ ,  $d_i \in \{2, 3, \dots, k\}$  and  $E[R] = E[R]^*$  to achieve the system reliability target. The  $r_i$  of the  $n - k$  harder to improve components are held constant.
- **DW.9** When  $k = n$ , and the system reliability target is not met,  $n - 1$  of  $n$  equations in the system are of

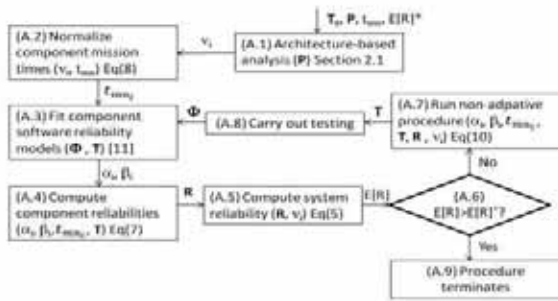


Figure 2. Non-adaptive procedure.

the form  $d_i = d_n$ ,  $d_i \in \{1, 2, \dots, n - 1\}$ , where  $d_n$  is the least responsive component. The  $n^{th}$  equation is  $E[R] = E[R]^*$ . Solving this system identifies the vector  $t_i$  that preserves the equality of  $d_i$  and makes  $E[R]$  equal the target reliability. Intuitively, this last scenario corresponds to finding the projected time allocation that achieves the target reliability while perfectly balancing the time adjusted reliability importance measures.

### 3.2 Iterative Adaptive Procedure

The Dale-Winterbottom algorithm produces the optimal testing time allocation vector  $\mathbf{T}^*$  in a single iteration only when the time/reliability functions of the components are known with complete certainty. Once the testing time is allocated, and the components are tested for the allocated time, additional faults may manifest during this testing process. The discovery of such additional faults will adversely alter component parameters increasing the estimated number of faults  $\alpha_i$  and decreasing the fault detection rate  $\beta_i$ . As a result, the reliability estimates of the components will drop due to which the components will need to be tested for an additional time. This necessitates an iterative test time allocation procedure that utilizes up to date fault detection data as it becomes available. The DW is used as a subroutine within the adaptive procedure. Upon termination, it returns a suggested time allocation vector to which is then used for component testing. The adaptive approach is designed to increase the resources allocated to a component depending on the failure behavior it exhibits during a specific testing period. The procedure accepts the DTMC representing the application architecture denoted by  $\mathbf{P}$ , total mission time  $t_{mn}$ , the target system reliability  $E[R]^*$ , the initial vector of testing times  $\mathbf{T}_0$ , and the vector of failure times for each component observed during these testing times  $\Phi_i$ . The steps involved in adaptive procedure are shown in Figure 2.

- **A.1** Compute the expected number of visits to each component using the matrix  $\mathbf{P}$ .

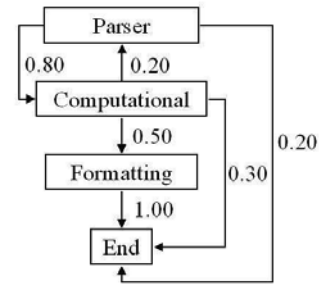


Figure 3. Architecture of European Space Agency Application

- **A.2** Estimate the component mission times  $t_{mn_i}$  using Equation (4).
- **A.3** Using the initial vector of testing times and observed failure behavior  $\Phi_i$ , estimate component parameters  $\alpha_i$  and  $\beta_i$ .
- **A.4** Using Equation (3), compute initial component reliability estimates,  $r_i$ .
- **A.5** Compute system reliability using Equation (1).
- **A.6** If  $E[R]$  exceeds the reliability target, the procedure terminates else go to **A.7**.
- **A.7** Invoke the DW procedure to determine a time allocation vector  $t_i$  that minimizes the total time needed to reach the system reliability target  $E[R]^*$ .
- **A.8** Test each component according to the time indicated by DW. If additional faults are discovered during testing, component parameters determined in (A.3) will be re-estimated using the additional failure data. The adaptive procedure terminates when no faults are discovered in any component during the testing applied during (A.8).

## 4 Illustration

In this section, we illustrate the adaptive reliability optimization procedure using an example application developed by the European Space Agency (ESA). The application consists of three components and its architecture is depicted in Figure 3. The mean number of visits to components are given by  $\nu_1 = 1.1905$ ,  $\nu_2 = 0.9524$ , and  $\nu_3 = 0.4762$ .

We consider the reliability objective of  $R^* = 0.8$ , and assume that each component has undergone initial testing for a time period of 10 units. The number of faults observed from each component are 12, 23 and 18 respectively, and using the observed failure behavior up to this time we estimate the component parameters using maximum likelihood method [5]. These parameters are summarized in

Table 1. Assuming a mission time of  $t_{mn} = 3$ , the normalized mission times of Equation (4) are  $t_{mn_1} = 1.3636$ ,  $t_{mn_2} = 1.0909$ , and  $t_{mn_3} = 0.5455$ .

**Table 1. Component parameters after initial testing**

Component 1	Component 2	Component 3
$\alpha_1 = 13.719$	$\alpha_2 = 26.257$	$\alpha_3 = 20.954$
$\beta_1 = 0.220$	$\beta_2 = 0.211$	$\beta_3 = 0.219$

#### 4.1 Iteration one

The reliabilities of the components computed using Equation (3) are 0.674, 0.521, and 0.768 respectively. The system reliability estimate computed using Equation (1) is 0.296. Since the system reliability estimate is less than the target, we now apply the Dale-Winterbottom method described in Section 3.1 to determine how testing time should be allocated to the components to achieve the reliability target.

Using the estimated parameters and component reliabilities, the time adjusted reliability importance computed using Equation (6) is  $\mathbf{D}_0^{DW} = (32.736, 25.711, 122.704)$ . Component two has the lowest time adjusted reliability importance, and hence, is the first target for improvement. Improving it to match the time-adjusted measure of component one boosts its reliability to 0.599, and the new time-adjusted importance measures are 28.648, 28.648, and 107.381 respectively. This boost in the reliability of component two will be achieved by allocating an additional 1.142 units of time, and the resulting system reliability would be 0.338. Since this testing time allocation does not meet the reliability requirement, we consider improvements to components one and two; with lowest time-adjusted reliability measures. DW suggests that the reliabilities of components one and two should be improved to 0.900 and 0.872 respectively, and the resulting time-adjusted measure of 53.174 will be the same for all three components. As the reliabilities of components one and two increase, the time-adjusted measure of component three decreases. This occurs because as components one and two are made more reliable, subsequent improvements to these two components will be costly, and hence, component three begins to emerge as an attractive improvement target. The reliabilities of components one and two is improved by allocating a total of 16.018 and 17.390 units of testing time to these components, bringing the system reliability to 0.683. Since this reliability is still lower than the target, we now consider simultaneous improvements to all the three components.

The final iteration of DW raises the time-adjusted importance measures to a common value, with the corresponding component reliabilities as 0.940, 0.923, and 0.857, which will achieve the target system reliability. The projected

time needed to reach these component reliabilities is 18.46, 19.30, and 12.55 respectively. This ends the DW portion of the procedure, and the control returns to the adaptive procedure. Table 2 summarizes the evolution of the testing time allocations, time-adjusted importance measures, and component reliabilities during the execution of the DW procedure.

**Table 2. Dale-Winterbottom procedure, iteration one.**

Iter.	Measure	Com#1	Com#2	Com#3	$R$
$0^{th}$	Time	10	10	10	0.296
	Importance	32.76	25.71	122.70	
	Reliability	0.674	0.521	0.768	
$1^{st}$	Time	10	11.142	10	0.338
	Importance	28.648	28.648	107.381	
	Reliability	0.674	0.599	0.768	
$2^{nd}$	Time	16.018	17.390	10	0.683
	Importance	53.174	53.174	53.174	
	Reliability	0.900	0.872	0.768	
$3^{rd}$	Time	18.463	19.929	12.455	0.800
	Importance	77.621	77.621	77.621	
	Reliability	0.940	0.923	0.857	

#### 4.2 Iteration two

In practice, the components will be tested for the times suggested by the DW procedure. For the sake of illustration, we simulate the fault detection processes of the components for their suggested times. At the end of these suggested times, the total number of faults detected from the three components are 17, 34 and 20 respectively. Because of the newly revealed faults, component parameters and their reliabilities need to be re-estimated and these revised estimates are summarized in Table 3.

**Table 3. Revised parameter estimates.**

Component 1	Component 2	Component 3
$\alpha_1 = 19.086$	$\alpha_2 = 37.090$	$\alpha_3 = 22.838$
$\beta_1 = 0.153$	$\beta_2 = 0.126$	$\beta_3 = 0.194$
$r_1 = 0.809$	$r_2 = 0.678$	$r_3 = 0.815$

While component reliabilities have improved, they could not reach the targets specified by the first iteration of the DW procedure (last row of Table 2. Consequently, the system reliability of 0.487 falls short of its target of 0.8. This shortfall occurs because of the discovery of new faults during the allocated testing time, which increases the amount of testing necessary to achieve the intended reliabilities. Thus, a second iteration of Dale-Winterbottom is performed with the updated parameters.

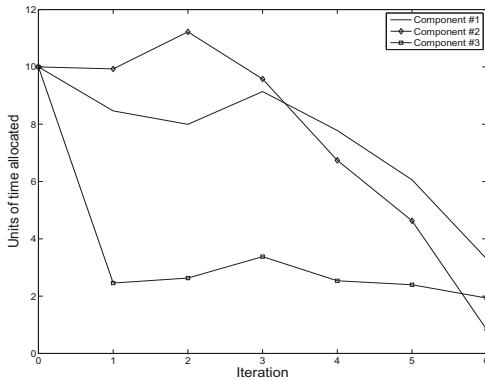


Figure 4. Time allocation by iteration

In the interest of space, we do not go through the step-wise execution of the DW procedure. The final result provided by the DW procedure suggests allocating a cumulative time of 26.458, 31.151, and 15.086 respectively. Once again, if no additional faults were detected during this allocated time, it would achieve the desired reliability target. Our simulation, however, indicates that additional faults will be detected during these times, bringing the total number of faults detected from each component to 21, 41, and 23 respectively. Again, the fault detection rate of each component drops due to the discovery of problems in each module, and the system reliability at the end of this iteration is 0.591 still short of the expectation.

### 4.3 Entire procedure

We now briefly summarize all iterations of the adaptive procedure (including the initial testing and the first iteration discussed above) that were necessary to achieve the desired reliability. Figure 4 depicts the amount of time allocated to each component during these successive iterations.

Component one receives increased resources during the third iteration, while component two is allocated more time during the second iteration. This is because the testing time allocated after the second and first iterations exposed eleven and four additional faults in these components respectively. When a cluster of faults is discovered, the reliability estimate drops and additional time is needed. This shows how the proposed algorithm adapts to the faults detected during testing, and alters the resources allocated in response.

Figure 5 shows the target and assessed reliabilities of component one after each iteration. The assessed reliability increases in each iteration, but the discovery of faults impedes it from achieving the desired target determined by iterations of the DW. The target and assess reliabilities are equal only in the final iteration when no faults are experienced during the time allocated to testing.

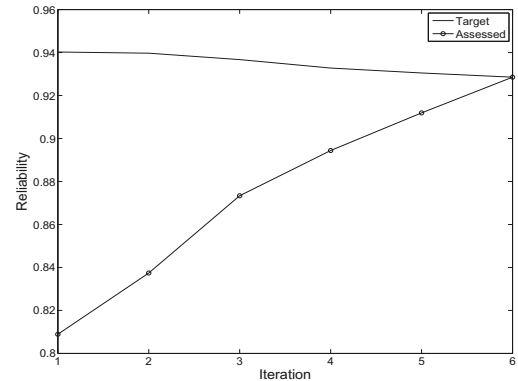


Figure 5. Target and assessed reliabilities in component one

## 5 Related research

Researchers agree that resource allocation guided by various factors [8] is important to reliability optimization. Earlier efforts considered software as a monolithic entity [7], and allocated resources to it as a whole. This approach proved inadequate as software became more complex and component-level resource allocation studies [2, 3] were proposed to address this shortcoming. Most of these studies ignored the architecture, treating components as non-interacting entities. These attempts made a limited impact because not all components contribute equally to application reliability. The next generation of models [10] considered architecture, but like previous efforts assumed components parameters were known in advance. A control theoretic model with multiple checkpoints [11] was developed, yet it treats software as a monolithic entity.

Our work pushes the state-of-the-art considering component based software reliability optimization in the context of its architecture. The approach eliminates a major drawback of many previous models, namely assuming knowledge of component failure parameters. By considering information as it becomes available, the method adaptively allocates time resources to components that will increase system reliability most efficiently.

## 6 Conclusion

In this paper, we present an adaptive procedure to minimize the time spent testing components of a modular software architecture. Unlike previous research, we do not assume advance knowledge of the component parameters. The conditional reliability expression of the Goel-Okumoto NHPP SRM describes the time/reliability relationship in each component. The Dale-Winterbottom method, a non adaptive iterative procedure, uses a time adjusted reliability

importance measure and serves as the core of our adaptive time allocation procedure. The illustration demonstrates the optimization process in detail, showing how it adapts to the discovery of clusters of faults in different components during different iterations.

There are several possible considerations for future research. The amount of time spent in each allocation cycle is not constant. A practical modification would be to make the time allocated in each period constant. This could facilitate synchronization with code reviews and other software process related activities that occur at regularly planned intervals.

## References

- [1] P. Kubat and H. Koch, "Managing Test-Procedures to Achieve Reliable Software," *IEEE Trans. Rel.*, vol. R-32, no. 3, pp. 299–303, aug 1983.
- [2] H. Ohtera and S. Yamada, "Optimal Allocation & Control Problems for Software-Testing Resources," *IEEE Trans. Rel.*, vol. 39, no. 2, pp. 171–176, jun 1990.
- [3] Y. Dai, M. Xie, K. Poh, and B. Yang, "Optimal Testing Resource Allocation with Genetic Algorithm for Modular Software Systems," *The Journal of Systems and Software*, vol. 66, pp. 47–55, 2003.
- [4] C. Dale and A. Winterbottom, "Optimal Allocation of Effort to Improve System Reliability," *IEEE Trans. Rel.*, vol. 35, no. 2, pp. 188–191, jun 1986.
- [5] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2<sup>nd</sup> ed. New York, NY: John Wiley & Sons, Inc., 2002.
- [6] S. Gokhale and K. Trivedi, "Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework," *IEEE Trans. Rel.*, vol. 55, no. 4, pp. 578–590, dec 2006.
- [7] A. Goel, "Software reliability models: Assumptions, limitations, and applicability," *IEEE Trans. Software Eng.*, vol. 11, no. 12, pp. 1411–1423, dec 1985.
- [8] W. Kuo, V. Prasad, F. Tillman, and C. Hwang, *Optimal Reliability Design: Fundamentals and Applications*. New York, NY: Cambridge University Press, 2001.
- [9] Z. Birnbaum, "On the Importance of Different Components in Multicomponent System," in *Multivariate Analysis II*, P. Krishnaiah, Ed. Dayton, OH: Academic Press, jun 1969, pp. 581–592.
- [10] J. Lo, S. Kuo, M. Lyu, and C. Huang, "Optimal Resource Allocation and Reliability Analysis for Component-Based Software Applications," in *Proc. COMPSAC-2002*, Oxford, England, aug 2002, p. 712.
- [11] J. Cangussu, R. DeCarlo, and A. Mathur, "A Formal Model of the Software Test Process," *IEEE Trans. Software Eng.*, vol. 28, no. 8, pp. 782–796, aug 2002.

# Enhancing Property Specification Tools With Validation Techniques

Salamah Salamah, Matthew Del Buono, Eric Baily, Sarah Printy, Derek Ferris, and Laurel Christian  
Embry-Riddle Aeronautical University, Daytona Beach, Florida  
salamahs, delbu9c1, baile6d4, print4a4, ferri91f, chris819@erau.edu

## Abstract

*Although formal approaches to software assurance such as runtime monitoring and model checking have been shown to improve system dependability, software development professionals have yet to adopt them. Among the reasons for this hesitance are the difficulty for clients and developers to write and validate formal specifications required by these approaches, and the lack of maturity of tools that can support the use of formal methods in a software development environment. This paper describes the Temporal Validator (TV) tool that assists in analyzing formulas in Linear Temporal Logic (LTL), which is one of the most widely used specification languages. The tool enhances the ability of users to validate generated formal specifications against their original intent in order to discover subtle errors. The paper also describes how the TV tool can be incorporated into the Property Specification (Prospec) tool to enhance users' ability to generate formal specifications that match his/her interpretations.*

## 1 Introduction

Formal approaches to software assurance require description of behavioral properties of the software system, generation of formal specifications for the identified properties, validation of the generated specifications, and verification of system's adherence to the formal specification. The effectiveness of the assurance approach depends on the quality of the formal specifications. A major impediment to the use of formal approaches in software development remains the difficulty associated with the development of *correct* formal specifications (i.e., ones that match the specifier's original intent) [5, 6].

Currently, there exists multiple formal specification languages that can be used in a variety of verification techniques and tools. Linear Temporal Logic (LTL) [10], Computational Tree Logic (CTL) [9], and Meta Event Definition Language (MEDL) [8] are some of these languages. The aforementioned languages can be used in a variety of veri-

fication techniques and tools. For example, the model checkers SPIN [7] and NuSMV [1] use LTL to specify properties of software and hardware systems. On the other hand, the SMV [2] model checker verifies system behaviors against formal properties in CTL. MEDL is used by JavaMac in runtime monitoring of java programs [8].

Formal languages have varying expressive powers, and as such, require a strong foundation in logic and mathematics for one to identify what can and cannot be expressed in a specific language. In addition, formal languages, by their nature, are hard to write, read, and validate. This problem is compounded if requirements must be specified in more than one formal language, which frequently is the case if more than one verification tool is used. Some research efforts continue to evolve to address the issue of automatic generation of formal specification by developers or customers who are not immersed in logic or the specification language of preference.

This paper introduces the Temporal Validator (TV) tool that can be used for validating formal specifications written in LTL. The tool was developed by students in the introductory course in Software Engineering at Embry-Riddle Aeronautical University in Daytona Beach. TV allows users to examine formal specifications (written in LTL) against traces of computations that represent different possible behaviors of the system being verified. The tool is intended to be part of the Property Specification (Prospec) tool [4, 12].

The paper is organized as follows; Section 2 provides a brief description of LTL and the Prospec tool. Section 3 provides the motivation of the work and the TV tool, while Section 4 introduces the tool and the ways it can be used. Finally, a summary of the work and future goals are presented.

## 2 Background: Linear Temporal Logic

Linear Temporal Logic (LTL) is a prominent formal specification language that is highly expressive and widely used in formal verification tools such as the model checkers SPIN [7] and NuSMV [1]. LTL is also used in the runtime verification of Java programs [17].



Formulas in LTL are constructed from elementary propositions and the usual Boolean operators *not*, *and*, *or*, *imply* (*neg*,  $\wedge$ ,  $\vee$ ,  $\rightarrow$ , respectively). In addition, LTL allows for the use of the temporal operators *next* ( $X$ ), *eventually* ( $F$ ), *always* ( $G$ ), *until*, ( $U$ ), *weak until* ( $W$ ), and *release* ( $R$ ).

Formulas in LTL assume discrete time, i.e., states  $s = 0, 1, 2, \dots$ . The meanings of the temporal operators are straightforward. The formula  $XP$  holds at state  $s$  if  $P$  holds at the next state  $s + 1$ .  $PUQ$  is true at state  $s$ , if there is a state  $s' \geq s$  at which  $Q$  is true and, if  $s'$  is such a state, then  $P$  is true at all states  $s_i$  for which  $s \leq s_i < s'$ . The formula  $FP$  is true at state  $s$  if  $P$  is true at some state  $s' \geq s$ . Finally, the formula  $GP$  holds at state  $s$  if  $P$  is true at all moments of time  $s' \geq s$ . Detailed description of LTL is provided by Manna et al. [10].

A major factor for the hesitance in using temporal logics in general is that they are hard to write. In addition, once specifications are written in LTL or CTL, it is hard to read and validate the meaning of the generated statement. For example, it is not immediately obvious that the LTL specification  $G(a \rightarrow F(p \wedge F(\neg p \wedge \neg a)))$  represents the English requirement “If a train is approaching(a), then it will be passing(p), and later it will be done passing with no train approaching”. The TV tool aims at providing the means by which developers and users can validate the meaning of such specifications as the one above.

## 2.1 Specification Pattern System and Prospec

Because of the difficulties associated with developing formal specifications, the Specification Pattern System (SPS) [3] developed a set of patterns to assist users in writing formal specifications in multiple formal languages. Patterns are high-level abstractions that provide descriptions of common properties that hold on a sequence of conditions or events in a finite state model. SPS patterns are grouped occurrence and order. Occurrence patterns are *universality*, *absence*, *existence*, and *bounded existence*. Order patterns are *precedence*, *response*, *chain of precedence* and *chain of response*. Chain patterns define a sequencing of events or conditions. Chain-precedence and chain-response patterns permit specifying a sequence of events or conditions as a parameter of precedence or response patterns, respectively. SPS allows the specification of sequences only to precedence and response patterns.

In SPS, a pattern is bounded by the scope of computation over which the pattern applies. The beginning and end of the scope are specified by the conditions or events that define the left (L) and right (R) boundaries, respectively. A study by Dwyer et. al. [3] identified the response pattern

as the most commonly used pattern, followed by the universality and absence patterns. These three patterns accounted for 80% of the 580 properties sampled in the study.

In many system properties multiple propositions may be needed to specify pattern or scope parameters. Mondragon et al. introduced Composite Propositions (CPs) [11] to handle pattern and scope parameters that represent multiple conditions or events. This was done as part of the Property Specification (Prospec) tool [4, 12]. The introduction of CPs supports the specification of concurrency, sequences, and non-consecutive sequential behavior on patterns and scopes. Mondragon proposes a taxonomy with twelve classes of CPs. In this taxonomy, each class defines a detailed structure for either concurrent or sequential behavior based on the types of relations that exist among a set of propositions. The complete list of CP classes and their LTL descriptions is available in Mondragon et. al. [11].

The original version of Prospec [12] is an automated tool that guides a user in the development of formal specifications. It includes patterns and scopes, and it uses decision trees to assist users in the selection of appropriate patterns and scopes for a given property. Prospec extends the capability of SPS by supporting the specification of CP classes for each parameter of a pattern or scope that is comprised of multiple conditions or events. By using CPs, a practitioner is directed to clarify requirements, which leads to reduced ambiguity and incompleteness of property specifications.

Prospec uses guided questions to distinguish the types of scope or relations among multiple conditions or events. By answering a series of questions, the practitioner is lead to consider different aspects of the property. A type of scope or CP class is identified at the end of guidance. The soon to be released Prospec 2.0 generates formal specifications in Future Interval Logic (FIL), Meta-Event Definition Language (MEDL), and LTL. The automatic generation of CTL specification is left as future work. More detailed description of Prospec 2.0 can be found in [4].

## 3 Motivation

Although, the soon to be released, Prospec 2.0 supports the generation of formal specifications in multiple languages including LTL, it currently does not provide sufficient support for validation of the generated properties. While Prospec and similar tools and approaches [3, 16] provide significant support for property specification, there is a real need to ensure that the generated formal specifications do, indeed, match the original intent of the specifier. Additionally, it has been shown that the specifications generated by these tools, do not always match the natural language description provided by these tools [15]. Such discrepancies are easier to find once it is possible to validate the specifications generated by these tools against a user’s defined

expected behavior of the system. Section 4.3 provides examples of how the TV can assist in this validation effort.

Providing the means to validate the generated specifications is extremely significant, as effective use of these formal specifications (whether in formal verification, design and code automation, or test cases development) is not possible if the generated specifications are faulty (i.e., do not match the developer’s original intent). Indeed, incorrect specifications could lead to the very mishaps their use is designed to prevent.

By their nature, formal specifications are hard to read and validate, and as such, support for validation and understanding of these specifications is required. For example, consider an ATM system with the following property: “The response to user approval of a withdrawal transaction includes: the user’s account is updated, money is dispensed, the receipt is printed, and the user’s ATM card is returned”. This property can be specified in LTL as follows: “ $G(\text{user\_approval} \rightarrow F(\text{account\_updated} \wedge X(F \text{ money\_dispensed} \wedge X(F \text{ receipt\_printed} \wedge X(F \text{ card\_returned}))))))$ ”. It is obvious that such a description is hard to validate by those stakeholders who are not immersed in LTL.

#### 4 Temporal Validator (TV) Tool

The Temporal Validator (TV) tool<sup>1</sup> allows for simple validation of formal specifications written in LTL. The tool allows users to test traces of computations that represent system behaviors against LTL specifications. A trace of computation is a sequence of states that depicts the propositions that hold in each state. The user models a state of computation by assigning truth values to the propositions of the LTL formula for a particular state. For example, a user may examine one or more combinations of the following: a proposition holds in the first state of execution, a proposition holds in the last state, a proposition holds in multiple states, a proposition holds in one state and not the next.

To clarify the above idea, consider the following LTL formula: “ $(F R) \rightarrow ((\neg R) U (P \wedge \neg R))$ ”. this formula specifies that “If  $R$  holds, then  $P$  must hold before  $R$ ”. Possible traces of computation to validate this formula are:

1. - - - -  $P$  - -  $R$  - -
2. - - -  $R$  - - -  $P$  - -

According to the English specification above, the first trace is consistent with the LTL formula while the second should fail. Note that in the traces above, the symbol “-” indicates that none of the propositions of interest is true in that particular state.

<sup>1</sup>The current TV tool can be requested from salamahs@erau.edu. Also a demo of the tool will be performed at the SEKE 09.

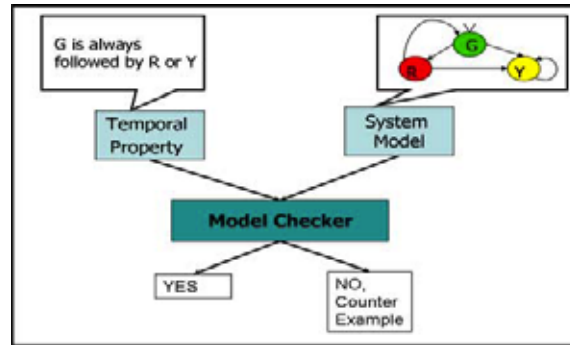


Figure 1. The Process of Model Checking

#### 4.1 Model Checking and TV’s General Approach

The idea behind the TV tool is to use the very formal verification technique that use the specification language as input to validate formulas written in that Language. Since LTL is widely used in model checking, TV makes use of the NuSMV model checker in performing the validation.

Model checking is a formal technique for verifying finite-state concurrent systems by examining the consistency of the system against system specifications for all possible executions of the system. The process of model checking consists of three tasks: modeling, specification, and verification.

**Modeling.** The modeling phase consists of converting the design into a formalism accepted by the model checker. In some cases, modeling is simply compiling the source code representing the design. In most cases, however, the limits of time and memory mean that additional abstraction is required to come up with a model that ignores irrelevant details. In NuSMV, the model is written in the SMV language [2].

**Specification.** As part of model checking a system, it is necessary to specify the system properties to be checked. Properties are usually expressed in a temporal logic. The use of temporal logic allows for reasoning about time, which becomes important in the case of reactive systems. In model checking, specifications are used to verify that the system satisfies the behavior expressed by the property.

**Verification.** Once the system model and properties are specified, the model checker verifies the consistency of the model and specification. The model checker relies on building a finite model of the system and then traversing the system model to verify that the specified properties hold in

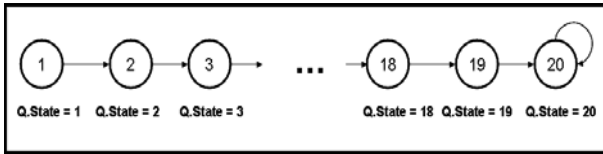


Figure 2. Model for LTL Validation

```

MODULE main
DEFINE
P : Q.State = 5
LTLSPEC F P
VAR
Q : seq();

MODULE seq()
VAR
State: 1..20;
ASSIGN
init(State):= 1;
next(State):= case
(State != 20) : {State + 1};
(State = 20) : {20};
esac;

```

Figure 3. SMV Code for LTL Validation

every execution of the model [2]. If there is an inconsistency between the model and the property being verified, a counter example, in form of execution trace, is provided to assist in identifying the source of the error. Figure 1 shows the process of model checking.

The general idea in TV is based on the work of Salamah et al., [15, 13, 14], and it consists of the following steps:

1. Create a simple model in SMV
2. map propositions in the formal specification to the variable(s) in the simple model,
3. run NuSMV with the model, formal specification, and proposition values as input, and
4. check for consistency.

The SMV model created for LTL validation, consists of a loop that starts with the value of the variable *state* equals 1 and continues to increment the value of states until it reaches 20, at which point it remains at 20. Figure 2 provides a graphical representation of the model, while Figure 3 provides the actual SMV code for the model.

While the details of the SMV code are irrelevant here, it is important to note that in each state of the model, the value of the variable *Q.State* changes to the value of the state. For example, in the first state, the value of *Q.State* is 1, it is 5 in the fifth state, and it is 20 in the last state. The importance of the value of the variable *Q.State* is that it is the value that propositions in the LTL formula are mapped to. For example, if one wants to validate the LTL formula “*FP*” (as in the fourth line in the SMV code), then the value of *P* has to be specified in terms of the variable *Q.State*. For example *P*

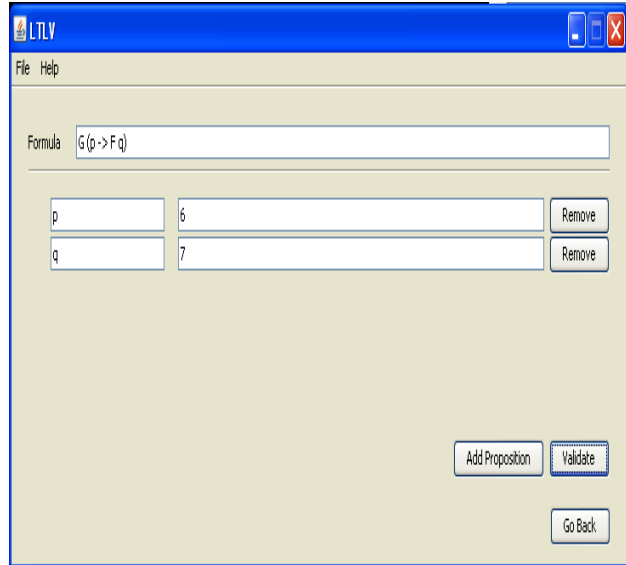


Figure 4. Specifying an LTL Formula and a Valid Trace in TV

can be specified as the truth value of the statement “*Q.State = 5*” (as in the third line in the SMV code), which is only true in the fifth state in the model.

#### 4.2 TV Interface

TV allows users to input the LTL formulas to be validated either by reading from an input file, or by manually entering the LTL formula and the trace of interest. Figure 4 shows the user input as the LTL formula  $G(p \rightarrow F q)$ . The formula specifies the *Response* property, i.e., “Anytime *p* holds, it must be followed by *q*”. In addition, the figure shows the user’s trace of computation as:

-----pq-----

Figure 5 shows the result of the verification (“Evaluates to TRUE”) as returned by NuSMV, as well as a visual description of the trace.

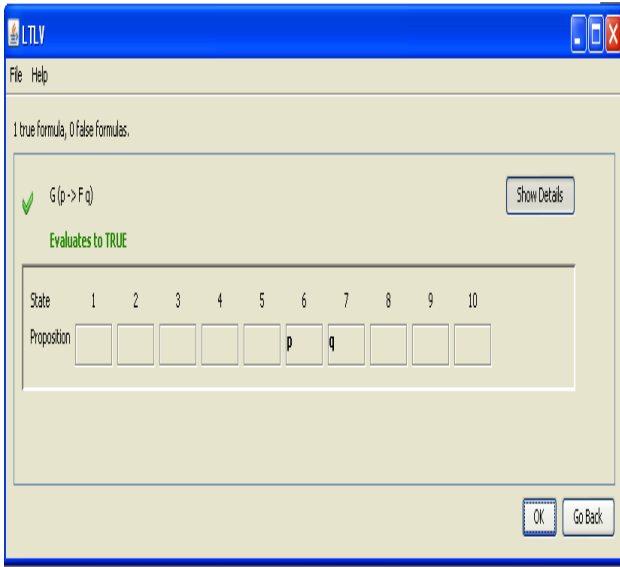
Figures 6 and 7 show the validation of the same LTL formula for *Response*, but with the trace:

-----p---q-----p-----

The result of the verification is FALSE, since *p* holds in the ninth state and *q* never holds after that. This is shown in Figure 7.

#### 4.3 Scenarios

The following scenarios illustrate the use of the TV tool to validate formal specifications.



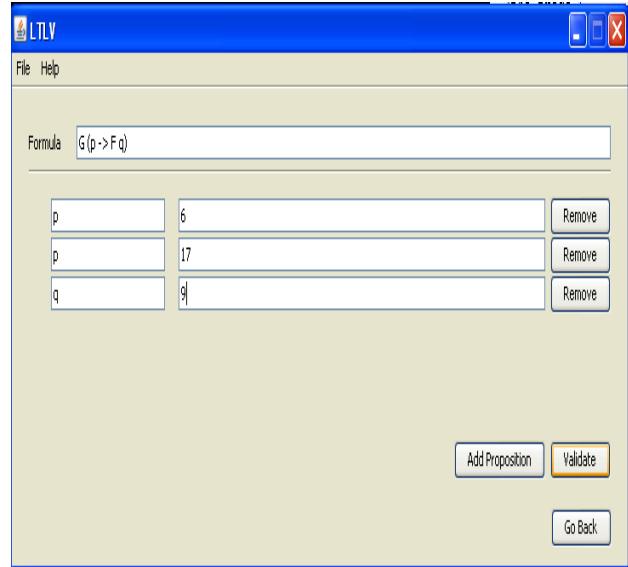
**Figure 5. Results of Validation by TV - Valid Result**

**Scenario 1.** Sarah is software engineer working on security issues in web services. She recognizes that the system must support the following requirement: “A message recipient shall reject messages containing invalid signatures, messages missing necessary claims, or messages whose claims have unacceptable values.” Since the project team will use a model checker to verify the algorithms, she needs an LTL specification. Sarah uses the Prospce tool and generates the following LTL formula  $G((invalid\_sign \vee miss\_claim \vee unacceptable\_value) \rightarrow F reject)$ .

Although Prospce allowed Sarah to generate the above LTL formula, being new to LTL, Sarah is most likely not sure if this formula actually depicts her original intent. So, she inputs the generated LTL formula into the TV tool and then tries to validate her understanding by testing multiple traces of computations against the formula. Sarah runs the following traces (symbol  $i$  stands for proposition  $invalid\_sign$ , symbol  $m$  for proposition  $miss\_claim$ , symbol  $u$  for proposition  $unacceptable\_value$ , and symbol  $r$  for proposition  $reject$ ):

- -- i --- r ----
- -- r - u -----
- u --- m -----
- (im) ----- r

Note that the last trace indicates that the propositions  $i$  and  $m$  hold in the same state (first state) of computation. Sarah observes the results from the TV tool and sees that



**Figure 6. Specifying an LTL Formula and an Invalid Trace in TV**

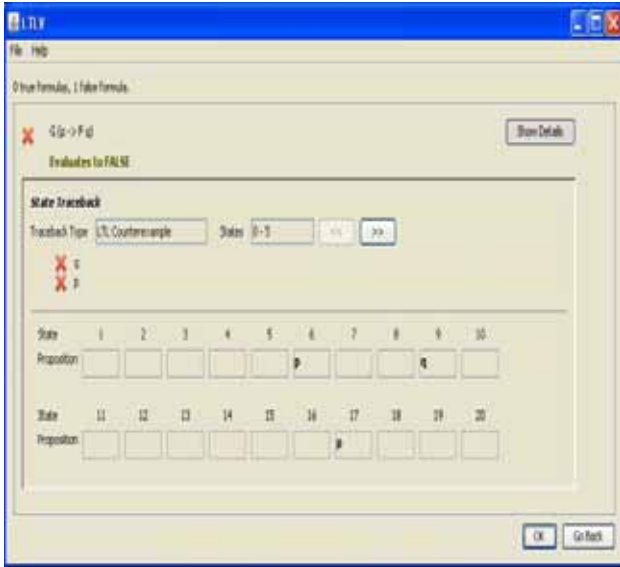
they match her expected results for each trace (first and last traces are accepted while the others are not).

**Scenario 2.** Eric is a student in Software Engineering. He was just introduced to Temporal Logics and was asked to write some specifications in LTL. One of the properties Eric is trying to specify is the following “Every request must be followed by an acknowledgment.” Eric starts by assigning the symbols “ $r$ ” and “ $a$ ” to the propositions “ $request$ ” and “ $acknowledgment$ ” respectively. Eric comes up with the following LTL formula “ $G(r \rightarrow F a)$ ”.

To make sure that his specification matches his understanding, Eric runs test the specification against the following traces (along with his expected results):

- -- r --- a ---- (valid)
- -- a - a ----- (valid)
- -- a - r - a - - r - (invalid)
- -- (ra) ----- (invalid)

Except for the last trace, TV returns the same result as Eric’s expected result. From this, Eric learns that the “Eventually” operator ( $F$ ) in LTL holds if the operand (in this case  $a$ ) holds in the current state. This is the case since the current state is part of the future in LTL. As a result of this testing, Eric now has to reconsider his understanding of the original property. He talks to his project teammates and instructor and comes to the conclusion that the desired property indicates that “ $acknowledgment$ ” must “*strictly*” follow the “ $request$ ” (i.e., it has to hold in a future state other



**Figure 7. Results of Validation by TV - Invalid Result**

than the current state. Eric changes his original LTL formula to “ $G(r \rightarrow X F a)$ ”. He runs all the previous traces using the TV tool. The tool returns validation results that match Eric’s expected results.

## 5 Summary and Future Work

The TV tool was developed by students in the introductory course in Software Engineering at Embry-Riddle Aeronautical University. The tool allows users to examine formal specifications against traces of computations that represent the different behaviors of the software system being verified. Providing the means to validate formal specifications is extremely significant, as effective use of these formal specifications (whether in formal verification, design and code automation, or test cases development) is not possible if the generated specifications are faulty (i.e., do not match the developer’s original intent). Indeed, incorrect specifications could lead to the very mishaps their use is designed to prevent.

TV can also be used as part of property elucidation as a way to distinguish accepted and unaccepted behaviors of the system under consideration. The tool can also be used in academic sittings in efforts relating to formal methods and formal specification [13].

Future work includes extending the tool to allow for validation of specifications written in CTL. The foundations of that work have already been implemented [14] and we plan to have it included in the future release of TV. Ultimately, we intend to integrate the TV tool into the Prospec tool and

to link both property specification and property validation efforts. Finally, an important goal of this work is allow for the automatic generation of traces of computations and the expected results (as in Section 5.3) for LTL and CTL specifications. This will relieve the user of the burden of defining those traces, as well as provide another way of validating specifications.

## References

- [1] Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M., “NuSMV: a new Symbolic Model Verifier” International Conference on Computer Aided Verification CAV, July 1999.
- [2] Clarke, E., Grumberg, O., and D. Peled. Model Checking. MIT Publishers, 1999.
- [3] Dwyer, M. B., Avrunin, G. S., and Corbett, J. C., “Patterns in Property Specification for Finite-State Verification,” Proceedings of the 21st Intl. Conference on Software Engineering, Los Angeles, CA, USA, 1999, 411-420.
- [4] Gallegos, A., Ochoa, O., Gates, A., Roach, S., Salamah, S., and Vela, C., “A Property Specification Tool for Generating Formal Specifications: Prospec 2.0”. In the Proceeding of the Conference of Software Engineering and Knowledge Engineering (SEKE), Los Angeles, CA, July 2008.
- [5] Hall, A., “Seven Myths of Formal Methods,” IEEE Software, September 1990, pp. 11-19.
- [6] Holloway, M., and Butler, R., “Impediments to Industrial Use of Formal Methods,” IEEE Computer, April 1996, pp. 25-26.
- [7] Holzmann, G. J., “The model checker SPIN” IEEE Transactions on Software Engineering., 23(5):279-295, May 1997.
- [8] Kim, M., Kannan, S., Lee, I., and Sokolsky, O., “Java-mac: a runtime assurance tool for java. In Proceedings of Runtime Verification (RV’01), volume 55 of Electronic Notes in Theoretical Computer Science. Elsevier Science, 2001.
- [9] Laroussinie, F. and Ph. Schnoebelen, “Specification in CTL+Past for verification in CTL,” Information and Computation, 2000, 236-263.
- [10] Manna, Z. and Pnueli, A., “Completing the Temporal Picture,” *Theoretical Computer Science*, 83(1), 1991, 97–130.
- [11] Mondragon, O. and Gates, A., “Supporting Elicitation and Specification of Software Properties through Patterns and Composite Propositions,” Intl. Journal Software Engineering and Knowledge Engineering, XS 14(1), Feb. 2004.
- [12] Mondragon, O., Gates, A., and Roach, S., “Prospec: Support for Elicitation and Formal Specification of Software Properties,” in Proceedings of Runtime Verification Workshop, ENTCS, 89(2), 2004.
- [13] Salamah, S., and Gates, A., “A Technique for Using Model Checkers to Teach Formal Specifications” Proceedings of the 21st IEEE-CS International Conference on Software Engineering Education and Training (CSEE&T), Charleston, SC, April 2008, 181-188.
- [14] Salamah, S., Gallegos, I., and Ochoa, O., “A Novel Approach for Software Property Validation” Proceedings of the International Conference for Software Engineering Theory and Practice (SETP), Orlando, FL, July 2008
- [15] Salamah, S., Gates, A., Roach, S., and Mondragon, O., “Verifying Pattern-Generated LTL Formulas: A Case Study. Proceedings of the 12th SPIN Workshop on Model Checking Software. San Francisco, California, August, 2005, 200-220
- [16] Smith, R.L., Avrunin, G.S., Clarke, L.A., and Osterweil, L.J. “PROPEL: an approach supporting property elucidation.” In Proceedings of the 24rd International Conference on Software Engineering. 2002, pp. 11-21
- [17] Stolz, V., and Bodden, E., “Temporal Assertions using AspectJ”, *Fifth Workshop on Runtime Verification* Jul. 2005.,

# Supporting Good Decision Making at Early Stage of Software Design

Hung-Fu Chang  
Computer Science Department  
University of Southern California  
Los Angeles, CA 90089, USA  
hungfuch@usc.edu

Stephen C-Y. Lu  
Viterbi School of Engineering  
University of Southern California  
Los Angeles, CA 90089, USA  
sclu@usc.edu

## Abstract

*Early stage decisions have more influence than the later stage decisions because they determine the major structure of the software and the difficulties of the overall system development. Due to the increasing demand of function and flexibilities of software design, if a poor decision has been made at the early stage, the resulting software becomes more complex than the designer expected. However, making a good design decision is not very easy especially at the early stage because of insufficient design details. To overcome this challenge, we use the Axiomatic Design theory to augment Object-oriented software design to propose a new approach - AD-based software design method. This paper explains the basic concepts of Axiomatic Design theory and how to use the AD-based software design method. We particularly focus on the benefit of the early stage decision evaluation that AD theory brings. To provide a comprehension, we use a case study to show how the designer improves the program structure by using the evaluation principle of the AD theory. The results show that, by using the AD-based software design method, the designer can explicitly evaluate the design at the early design stage and maintain the good decision along the design process. This method can lead to better software systems with higher quality.*

**Key Words-** Software Conceptual Design, Decision-Making, Software Dependency, Axiomatic Design Theory

## 1. Introduction

In all engineering domains, “design” is always a goal-oriented decision-making process which satisfies from the need – what to build to how to build it and end up with a technical system [1]. This decision-making process plays a central role during the system design. If the design decision at every design step can

be carefully made, a good foundation for the later stage implementation can be set.

The overall design process contains two primary design stages which are conceptual and detailed design stages. It is commonly believed that early stage (i.e., conceptual design stage) decision-making has bigger influences than the later stage. However, making a good decision at the conceptual stage is not easy because the designer has less information at this stage. In addition, even though the designer makes a good design decision in the beginning of the design, follow-up design decisions may not be consistent with the early good decision. Especially, due to the ultimate flexibilities of the software development and the demand for more function for software [6], once the designer makes a poor design decision, this deficiency leads to complex dependencies between modules that not only brings the challenges of the implementation but also may reduce the quality of the resulting system.

To remedy this deficiency, we employ an axiom-based theory, called Axiomatic Design (AD) that was developed by N. P. Suh in the late 1970s [5]. In addition to capturing abstraction in layers, the AD theory adds a second dimension, called “domain”, to represent the different types of design decisions. Regardless the application, four generic design domains, customer, functional, physical, and process domains are introduced. With this two-dimensional domain-and-layer structure, the theory employs the horizontal mapping operations across two adjacent domains and the vertical decomposition operation across two adjacent layers. During the early design stage, this two dimensional operation helps the designer to generate and to map two hierarchies in functional and physical domains. As well, in these two adjacent domains, a design equation is built to show the dependency between these two types of design decisions. Mostly importantly, the AD theory provides two fundamental design axioms, namely the Independence Axiom and the Information Axiom, as evaluation criteria to assist the designer to compare and select those design concepts that minimize

dependencies among design decisions which lead to the simplest design. These key concepts of the AD theory are domain-generic and can offer the software designer capability to evaluate the design decision at the early stage. In this way, the software design practice can be improved.

In the rest of paper, we introduce the basic concepts of the AD theory; particularly, we focus on how to evaluate the early stage decision-making and maintain the consistent decision through the design. We also propose an AD-based software design method that complements OO approach and explain how this method can work with a step-by-step procedure. We finally apply our AD-based software design method to a software project, describe how the AD-based method can benefit the early stage software design and improve the design result.

## 2. Apply Axiomatic Design Theory in Software Development

### 2.1. Key Concepts of AD Theory

The nutshell of the AD theory has the following four key elements:

- **Domains:** design is viewed as a continuous mapping between two adjacent domains, i.e., from the upstream “what” domain to the downstream “how” domain,
- **Hierarchies:** hierarchies represent “design architecture” created by successive specializations from top to bottom levels (layers) of details of design concepts,
- **Zigzagging:** alternating mapping-specialization between adjacent domains and layers, constrained by design choices made at the higher layer of downstream domains, to continue from higher-level design concepts into lower-level ones,
- **Axioms:** employ two fundamental axioms, the Independence Axiom and the Information Axiom, to guide the creation, comparison and selection of good design concepts during zigzagging.

Domain is a new concept introduced by the AD theory to categorize the different types of design decisions (i.e., “what” vs. “how”). Unlike traditional approaches which are mostly based on one-dimensional decomposition (which mix-up what and how into a single hierarchy), the zigzagging process (i.e., mapping across domains then specializing into layers) in the AD theory provides a two-dimensional roadmap to guide design decisions. Regardless applications, four generic domains can be identified in any design task:

1. The Customer Need (CN) domain, which captures the desired features of the technical systems and their expected performances that customers care and/or call for.

2. The Functional Requirement (FR) domain, which is a minimum set of independent requirements that completely characterize/satisfy the targeted CNs.

3. The Design Parameter (DP) domain, which is a set of key physical parameters that instantiate the design concepts to satisfy the chosen FRs.

4. The Process Variable (PV) domain, which is a set of key variables (and specific values) which specify the physical process and/or resources required for implementing the above specified DPs.

The Axiomatic Design Framework is shown in Figure 1.

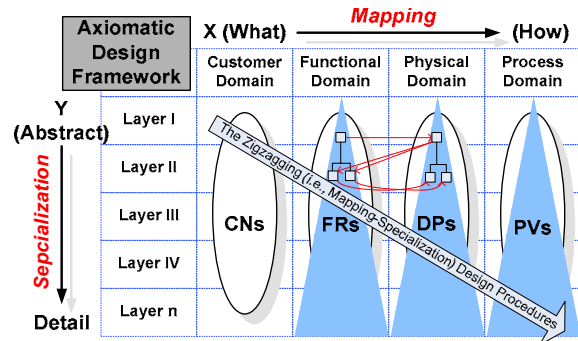


Figure 1. Zigzagging across Four Design Domains

During the zigzagging maneuvers (see Fig. 1), two directional operations are performed. The horizontal “mapping” operation (i.e., “zig” operation) creates the “means-of” dependencies between functional and physical domains. The vertical “specialization” operation within the domain creates the “part-of” hierarchy; however, this vertical operation should also be guided by information from the upper layers from the adjacent domain (e.g., from DP to FR). Such a guided mapping from upper level in adjacent domain to the down level is the “zag” operation. Additional design “constraints” (that don’t have to be independent of each other) along the zag process often are added to represent “bounds” on acceptable design solutions to guide the zigzagging process. After the zigzagging process, two hierarchical structures are created in the functional and physical domains, respectively.

The AD theory proposes that the simplest design should be the ideal design. To achieve this, two design axioms must be maintained through the design process and they are the Independence Axiom and the Information Axiom. The Independence Axiom states that each chosen FR must be independently satisfied by

a DP. The Information Axiom states that among those design concepts (i.e., the FR/DP mappings) that satisfied the first axiom, the one with the least information content should be chosen. At each level of the decomposition, the mapping between the functional and physical domains can form a design equation  $\{FR\} = [A]\{DP\}$ , where  $[A]$  is the Design Matrix  $[A]$ . This matrix can be used by the designer to investigate whether the Independence Axiom has been satisfied during the zigzagging process.

## 2.2. Uncoupled, Decoupled and Coupled Design of AD Theory

The key to the possibility of separating a good design from a bad one at early stages is the ability to distinguish the dependencies of design decisions in the functional and physical domains. From the perspective of the AD theory, a design that can fully satisfy the independence of the FR at the earlier design stage should be a good design. To lead to good designs, the Design Matrix  $[A]$  that denotes a dependency between FRs and DPs can help the designer to examine the functional independence (or couple) of the design. According to the characteristics of the  $[A]$ , three kinds of designs, uncoupled, decoupled and coupled, can be found. If all the non-zero elements in  $[A]$  are only in the diagonal position, it is an uncoupled design. The decoupled design is identified while all the elements in the upper triangular of  $[A]$  are equal to zero. However, when  $[A]$  is not the uncoupled or decoupled case, it must be a coupled design. The uncoupled design means that each FR is only satisfied by one corresponding DP; for example, in Fig. 2,  $FR_1$  is only related to  $DP_1$  (i.e.,  $FR_1 = A_{11}DP_1$ ). The FR in the decoupled design is not completely independent. However, AD theory believes the independence of the FR can be identified while the DPs can be determined in a particular sequence. As a result, in terms of good design, except uncoupled designs, the decoupled design is also acceptable.

$$\begin{array}{l}
 \text{Uncoupled Design} \\
 \left. \begin{array}{l} FR_1 \\ FR_2 \\ FR_3 \end{array} \right\} = \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \end{bmatrix} \begin{array}{l} DP_1 \\ DP_2 \\ DP_3 \end{array} \\
 \\
 \text{Decoupled Design} \\
 \left. \begin{array}{l} FR_1 \\ FR_2 \\ FR_3 \end{array} \right\} = \begin{bmatrix} A_{11} & 0 & 0 \\ A_{21} & A_{22} & 0 \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{array}{l} DP_1 \\ DP_2 \\ DP_3 \end{array}
 \end{array}$$

Figure 2. 3X3 Design Matrix of Uncoupled and Decoupled Designs

The  $[A]$  at each level is also an aid for designers to check if the sub-level design choice is consistent with the super-level design intention. By examining the  $[A]$

at each level, once the choice that the designer states at the sub-level violates the super-level one, a redesign (re-create the structure) should be made instead of continuing it. Therefore, through such a process, a good design at the earlier stage can be guaranteed.

## 2.3. AD-based Software Design Method

To fully utilize the benefits of the AD theory, it is necessary to know how to apply the AD theory to software design. Our goal is to use the AD theory to complement Object-oriented (OO) design; therefore, the good software design at the early stage can be determined, which finally leads to a better OO software system.

However, the fundamental differences between physical and software engineering makes the AD theory difficult to be understood in the software community. Software is intangible; unlike physical systems, software structure that is composed of various programs is dimensionless without physical and/or geometric constraints. In the design of physical systems, functional requirements and their corresponding expected behaviors (i.e., FR decisions) are the designer's intention. They are conceptually described and created based on customer needs. The DP decisions that are the design consequences of the FR result in the actual behaviors. In physical systems, the expected behaviors (or functional requirements) are always different from the resultant behaviors; therefore, the separation between FR and DP is very clear. This separation however is implicit in software systems because they are intangible in the physical world. For example, Booch's OO analysis and design method does not explicitly distinguish the design intentions (i.e., functional requirements or expected behaviors) and the system specifications (i.e., result behaviors).

In software design, each FR also represents the designer's choice or intention to satisfy customer needs, which can be transformed into system's or component's "expected" behaviors. However, the DP that realizes the FR forms a hierarchy that holds the "part-of" relationship in the physical domain. This hierarchy structure lays out the program static structure that the designer creates.

Each step of our proposed AD-based software design method is described below.

- **Elicit Customer Needs**

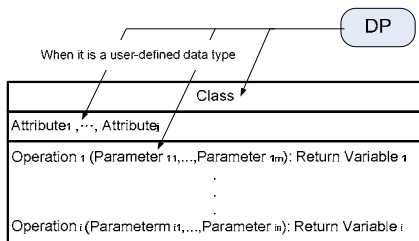
Requirement descriptions of the desired software systems are collected by interviews with clients. To ascertain the client's actual needs rather than what the designer think, a requirement elicitation technique is



often applied. Then, those real customer needs (CNs) are listed in the customer domain.

- **Specify FRs and DPs through Zigzagging Process**

At this stage, developers lay out the FRs and the DPs through the “zigzagging” process. All the FRs are based the CNs. The root level represents the most abstract function, contains no detailed information and shows the design intent. At the leaf level, the designer has to identify all the objects; thus, the composition of the objects and the instances that is passed between objects both need to be clearly allocated. Since the class describes the structure of the object, the designer can transform the “part-of” hierarchy (rather than inheritance relationship) into the class diagram in order to show the static relationship between objects. More specifically, attributes or parameters made of user-defined data types (classes) are identified (see the class diagram representation in Fig. 3). Between these levels, components or higher level objects (which compose lower level objects) are chosen (through the specialization operations) to compose the overall architecture. During this specialization, the designer must always consider the associated interfaces and names. As well, additional constraints are often identified through the “zag” process from upper level DPs to its sub-level FRs. This type of constraint limits the designer’s choice in the sub levels. For example, if the upper level DP decides to use three-tier structure and the third tier is a database structure, the sub-level FR and its mapping DP could be limited to choose a database protocol. Moreover, the FR decomposition is very different from the traditional functional decomposition in the structure software design. The FRs do not concern the sequence of the operation or any data flow. The FRs are the functionalities that the system or the component has need to perform. Therefore, if the designer needs, they can even apply some technique like the use case analysis to specify the functionalities of the system.



**Figure 3. Class Diagram for Showing the Possible DP Sources in the Object**

Evaluation of the design equation (or Design Matrix [A]) is also handled during this zigzagging process. At

each level of the decomposition, two exams have to be done using the design equation. First, the lower level design decision needs to be consistent with the upper level ones. Second, the designer evaluates that the design is an acceptable one (i.e., the design is either uncoupled or decoupled). If one of them is violated, the designer has to do the design again.

- **Model from Other Viewpoint**

If the static structure representation is not enough for the designer to express the design result or to understand the functional coupling, they can plot the internal relationship between modules at different viewpoints. For example, the sequence diagram is one of the UML diagrams in OO design to show the dynamic message between objects. Other representations like inheritance relationships are also extracted at this time.

- **Specify PVs from DPs according to Platform-Dependent Technologies**

The definition of the PV in AD theory is related to the manufacturing process. Therefore, in software development, the PV in the process domain is the code implementation that associates with a corresponding DP. While developers specify PVs through DPs, the technology they used must be considered such as the operation system or the programming language. Since our focus in this paper is to evaluate a good design at the early stage, we do not address any issues regarding this step.

### 3. Case Study of the AD-based Software Design Method

This section explains how we use the AD theory to complement OO design to achieve a good design at the early stage. To briefly demonstrate the maintenance of a good design solution, only a portion of the whole system design is shown.

#### 3.1. The Software Design Project

Our case study project is to build an application that has to generate specialized format script, called LoadRunner, according to a template. Figure 4 shows the requirements of the system. In the system, the software program needs to take multiple XML files as inputs and a single template file as specified by the user. Through the system’s processing, the outputs, multiple LoadRunner scripts are generated by combining XML elements with the template and are saved in the user-specified folder. In order to fit different operating systems and usages, the system requires a graphical user interface (GUI) and command

line manipulation running on different platforms. After all customer requirements are clearly captured and defined after several interviews with our clients, they are listed in the customer (CN) domain [10].

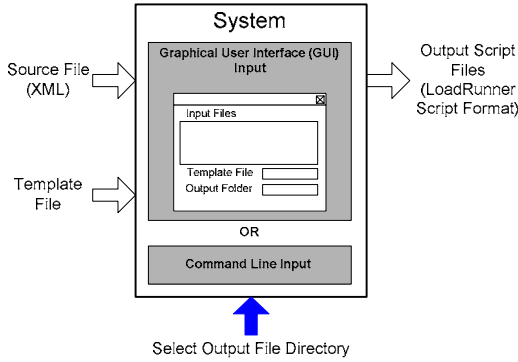


Figure 4. The System Requirement Description

### 3.2. The AD-based Design Result

A maximum of five specialization layers are created in both functional and physical hierarchies. A total of 26 objects (not the total number of DP nodes) are identified in the system.

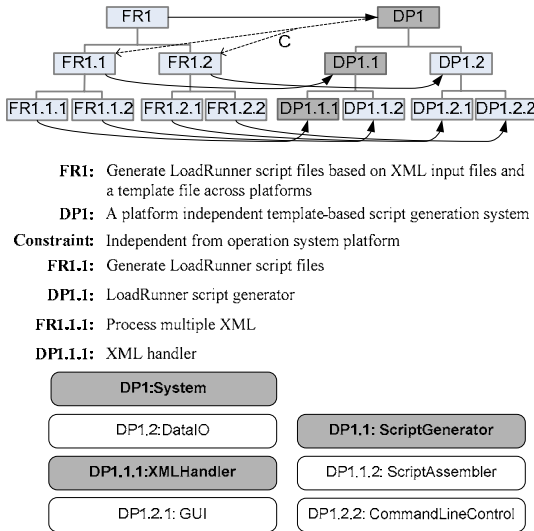


Figure 5. The First Three Level of Specialization

Figure 5 shows the first three levels of specialization result that includes the mapping from the FRs to the DPs (i.e., horizontal direction), and the name of the entity in the physical domain (i.e., the rounded rectangular). We only identify one constraint during the design. To briefly discuss how the design equation can assist the designer to make good decisions, only the

branch that contains DP1 to DP1.1.4 (i.e., the gray node in the Fig. 5 and Fig. 6) is selected for the demonstration.

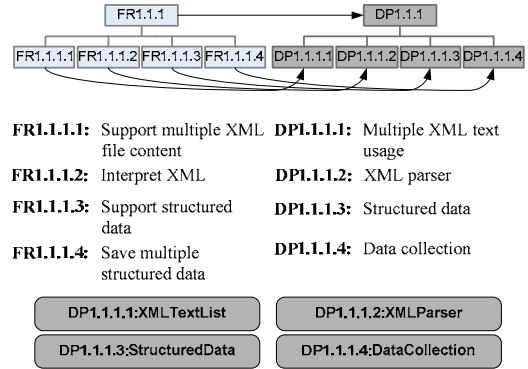


Figure 6. The Fourth Level of Specialization

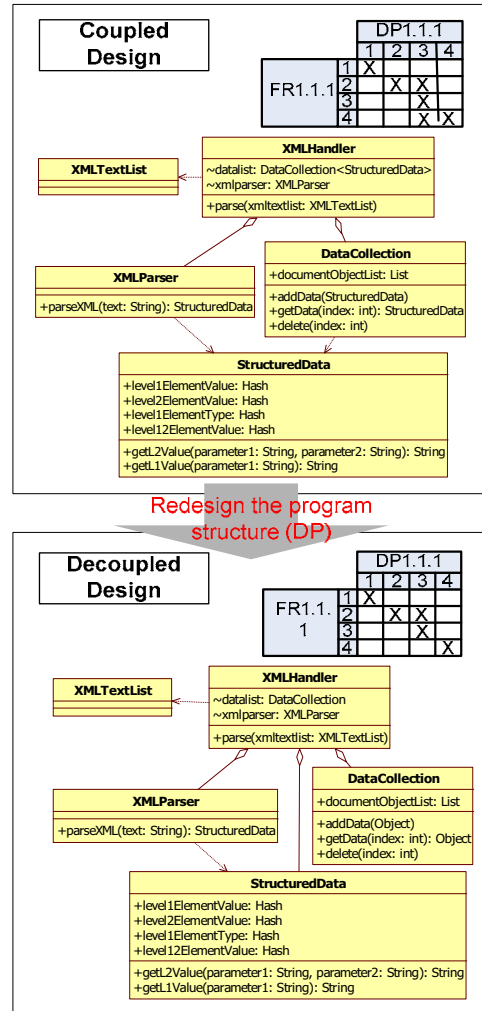


Figure 7. Class Diagram under DP 1.1.1

Under DP1.1.1 (XMLHandler), all the leaves are objects; therefore, the interfaces, parameters and attributes of the object are considered (see Fig. 6). The dependencies between objects can be shown in the class diagram and the design equation (see Fig. 7).

After we use the design equation that is built at every level of hierarchy to evaluate the design, the DP decisions at the first three levels are consistent and created as an uncoupled design. When more details are considered below the third level, the design equation shows the inconsistencies with the upper levels. In the first round design, we create a coupled design (see Fig. 7). The DP1.1.1.3 (StructuredData) has the dependencies with DP1.1.1.2 (XMLParser) and DP1.1.1.4 (DataCollection) because both objects use StructuredData as a return data type in their method. However, according to the design decision FR1.1.1.4, it is not necessary to use StructuredData as a return data type in DataCollection because DataCollection is only responsible for saving objects regardless of the object type. As a result, in the second round design at this level, we modify the DataCollection and remove the dependency between DataCollection and StructuredData. The DP structure at the fourth level becomes an acceptable decoupled design (by AD theory definition). The design modification procedure shows how the designer can change the program structure (or software architecture) based on the evaluation of the design equation at the early design stage.

#### 4. Conclusion

The decision-making in the early stage plays a more important role than it does in the rest of the development activities. A poor software design decision that is made at the conceptual period brings the overall difficulties to the software system. Especially, due to the complexity and the flexibility of the software design, having better design supports can reduce the complexity of their design decisions. Our AD-based software design method indeed offers a solution to the need. Using the design equation (Design Matrix [A]) with evaluation principle (i.e., uncoupled, decoupled and coupled design) to judge a good decision at each level becomes very easy in our method. In this way, the designer can improve the quality by maintaining the quality of the design decision according to the same criteria in the whole development. Those benefits are hardly found in any traditional software design method.

In the case study, by showing one part of the design result of the software project, we explained how the

designer uses the design equation to evaluate the design decision. If the decision is poor, the designer has to redesign it to create a better architecture; that is, the structure has less dependency between modules (programs) and is expected to form a simpler software system. The demonstration shows those benefits of using our AD-based software design method and helps the designer to make the decision during the software system design.

#### 5. References

- [1] Hubka, V. and Eder, W.E.. *Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge*, London: Springer-Verlag. 1996.
- [2] Lu, S. C-Y. and Suh, N.P., "Complexity in Design of Technical Systems", *Annals of the CIRP*, Vol. 5, 8/1/2009, pp. 241-245, 2009 (to appear).
- [3] Remco C. de Boer and Hans van Vliet, "On the Similarity between Requirements and Architecture", *Journal of Systems and Software*, p.544-550, November 2008.
- [4] Philippe Kruchten, "An Ontology of Architectural Design Decisions in Software-Intensive System", *2nd Workshop on Software Variability Management*, Groningen, NL, Rijksuniversiteit Groningen, 2004.
- [5] Suh, N.P., *Axiomatic Design: Advances and Applications*. Oxford University Press. 2001.
- [6] Grady Booch, Robert A. Maksimchuk, Michael W. Engel, Bobbi J. Young, Jim Conallen and Kelli A. Houston, *Object-Oriented Analysis and Design with Applications. 3rd ed*, Addison-Wesley. 2007.
- [7] M. Moore, R. Kazman, M. Klein and J. Asundi, "Quantifying the Value of Architecture Design Decisions: Lessons from the Field", *Proceedings of the 25th International Conference on Software Engineering*, Portland, Oregon, May 2003.
- [8] David L. Parnas, Paul C. Clements and David M. Weiss, "The modular structure of complex systems", *IEEE Transactions on Software Engineering*. Vol. SE-11. no. 3. March 1985.
- [9] *Software and System Requirement Definition of CodeGenerator Project*. [http://greenbay.usc.edu/csci577/spring2006/projects/team10/AsBuilt/SSRD\\_As-built\\_Final\\_S06b\\_T10.pdf](http://greenbay.usc.edu/csci577/spring2006/projects/team10/AsBuilt/SSRD_As-built_Final_S06b_T10.pdf)

# A Language for Modeling Software Development Life Cycles

Dr. Ernest Cachia, Mr. Mark Micallef  
Software Engineering Process Improvement Research Group  
Department of Computer Science  
University of Malta  
ernest.cachia@um.edu.mt, mmica01@um.edu.mt

## Abstract

*As the profession of software engineering has continued to mature, a substantial body of work has built up in the area of software development life cycles. A variety of generic models ranging from waterfall to agile have appeared over the past few decades, all claiming to address prominent issues in software engineering.*

*In this paper, the authors make an argument for developing a unifying language with the capabilities of defining and modeling any software development life cycle. The concept is similar to the way UML is a language that can be used to model any conceivable object oriented software solution. A number of reasons are provided as to why this would be a useful contribution to software engineering. The authors also discuss work which they have been carrying out in this area and outline the main features and semantics of such a language.*

## 1 Introduction and Motivation

Software development life cycles have been around for decades. Over the years, they have evolved so as to cater for different circumstances, system types, organisational structures, time scales and so on. When Royce proposed what was to (somewhat mistakenly) become known as the waterfall development life cycle in 1970[1], he attempted to propose a structured approach to software development. However, by his own admission, the ideas in the article were personal views and prejudiced by his own personal experiences[1]. If one reads articles about more recent approaches such as agile practices [2] [3], they are based on a number of conceived best practices which have been observed to work in projects with particular characteristics. As such, software development life cycles tend not to have a scientific basis to them. To date, there is no way to 'prove' that if one set of practices is followed in a development

project a particular outcome will occur. Indeed, one might argue that it is unrealistic to expect otherwise.

The authors of this paper argue that the state of development process management today presents practitioners with a number of problems. Firstly, it is difficult to formally reason about development life cycles. Since they are effectively a number of best practices wrapped up in stages and work flows, it is not a straight forward task to fully comprehend what effects a particular life cycle will have on one's projects. There have been attempts at providing guidelines for practitioners in this area, most of which seem to occur in the industrial sector. One such attempt is the D<sup>3</sup> cube [4] which recommends a development life cycle based on three factors: the project's *time scale*, *budget* and *stability of functional requirements*. However, these are not the only factors which should be taken into account when choosing a development life cycle. Arguably, development life cycles are an over simplification of what is in reality, a truly complex task. Consider the following statements: "This project needs to produce a highly secure product". "Our company will reduce its workforce by 10% by the time the project ends". "This website needs to be available in 50 languages". These are examples of typical real-life situations which will effect the way a product is developed. In most cases, an organisation will need to mould its life cycle to its particular needs.

Another problem with life cycles today is the lack of prediction and modeling capabilities. It is difficult to give an accurate estimate of how long a project will take, what will be delivered and what level of quality deliverables will exhibit. Even with meticulous planning, it is inevitable that projects will come up against unplanned situations which hamper progress. It would be useful to have a way of modeling a development process such that scenario tests could be carried out on it prior to project commencement. For example, if there is a risk that a 3<sup>rd</sup>-party vendor will not deliver on time, how would the development process

handle it? Similarly, a model could help in selecting corrective measures when something has gone wrong. If there are multiple ways to approach the solution of a problem, a model could help select the best one.

Finally, there is no common way to represent and compare different development life cycles. Life cycles are usually proposed in articles or papers as textual descriptions supported by diagrams to help clarify the author's ideas. It would be useful to have a common way of representing development life cycles so as to make it easier to compare and reason about them.

In this paper, the authors are proposing a language for modeling software development life cycles. It is claimed that such a language will address the issues discussed here and also give the management of software development life cycles a more scientific basis from which to operate.

### 1.1 A note on scope

The work discussed here refers to a language definition which has not yet been fully defined. The authors are using this paper as a platform for sharing their research ideas and progress made thus far so as to be able to incorporate feedback into a final version of the language being proposed here. Consequently, the scope of this paper is to make a case for the development of a life cycle modeling language and highlight key features, semantics and mechanisms within such a language. It is the goal of the authors to publish a full specification of the language within the next few months. As one may appreciate, it is difficult to publish a language specification within the constraints of a conference paper.

## 2 Desirable Language Characteristics

This section discusses what desirable characteristics a development life cycle modeling language should exhibit. Knowing these properties will help shape the language itself as well as provide a measuring stick for analysing the language.

As an initial requirement, it would be desirable if the language was *sufficiently expressive* to model any software development life cycle, be it an standard or one which is customised to a particular organisation. As such, it would also be advantageous if the language were *extensible*. It is proposed that the language aims to explicitly model the 'lowest common denominator' characteristics of software development life cycles whilst providing a way to allow practitioners to plug in any other concepts. That is to say that it should not seek to explicitly provide syntax and semantics for concepts particular to any existing life cycles.

Rather, it should provide syntax for generic concepts which apply across all life cycles. Therefore syntax will be provided for a concept such as a life cycle *stage* but not for the agile-specific activity of *pair programming*. This makes sense because it is impractical to implement all known life cycle activities, concepts and their variants. Also, this approach will make it possible for the future integration of concepts which may have not yet been invented without the need to change the language itself.

Another desirable characteristic is for the language to be *easy to learn and intuitive* to use yet formal enough to provide analytical capabilities to practitioners. This will allow intuitive creation of models which could easily be understood by all stakeholders with minimal training. The language should also facilitate reasoning a deduction capabilities in order to allow users to carry out scenario testing on any models they build.

Finally, a user would typically want to quickly build a coarse model in order to get started and then refine appropriately over a period of time. Therefore, it would be desirable for the language to allow for varying levels of abstraction within models.

The next section will look into generic characteristics of software development life cycles in order to clarify decisions which have been made with regards to the structure of the language presented here.

## 3 Software Development Life Cycles

Prior to proposing a language to model software development life cycle, it is useful to analyse the entity being modeled with a view to identify generic characteristics about software development life cycles. Although all life cycles propose different activities and work flows, there are a number of characteristics which are common across all development life cycles. As discussed in section 2, the proposed language aims to explicitly model the 'lowest common denominator' characteristics whilst providing a way to allow practitioners to plug in any life cycle-specific concepts.

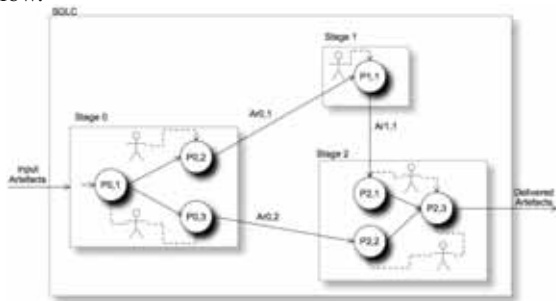
Having surveyed a number of life cycles [1][5][6][7][8], the authors propose the following unifying view. A software development life cycle is an entity which accepts inputs from external entities and ultimately delivers a number of artefacts as outputs. All software development life cycles have the concept of *stages*. A stage, sometimes also referred to as a *phase* is a coherent body of activities targeted towards completing one or more tasks which usually fall into one genre (e.g. design, coding, testing, and

so on). Most tasks involve the production of one or more artefacts but this need not always be the case.

Stages are composed of a one or more *processes*. These are also coherent bodies of work contribute to the achievement of their parent stage's goals. Processes may also produce artefacts as deliverables. Strictly speaking, the concept of stages is not essential because they usually act as a wrapper around a number of processes. Removing this wrapper should not alter the overall workings of a development life cycle. However, the concept of stages provides a useful abstraction tool which makes it easier for practitioners to model and understand development life cycles.

Processes are enabled through *actors*. The term *actor* refers to any entity which contributes work to the project. Actors could refer to actual human entities such as *Joe* or *Jane*. They could also refer to roles without the need to specify the individual who fulfills that role. Examples of role actors include *client*, *test analyst* and *review team*. Finally, an actor could also refer to non-human entities which somehow contribute to the process. A typical example of this might be a software system provides a service within the process.

The concepts discussed here are depicted in the figure below.



## 4 Language Definition

This section explains the core concepts, rules and semantics behind the proposed language.

### 4.1 Overview

The proposed language has its foundations in second-order logic and models life cycles which fit into the generic view discussed in section 3. The decision to base the language on second-order logic was made because at the end of the day, life cycle modeling will turn out to be a knowledge-base problem. That is to say, one will store knowledge about development life cycles, resources, circumstances and so on with a view to eventually query that

knowledge and use it for modeling purposes. Second-order logic is ideal for this kind of work and forms the basis of deduction-based knowledge management systems.

The core vocabulary of the language is be defined as follows:

$$SDLC := \{C, S, Ar\}$$

$$S := \{C, Ar, P\}$$

$$P := \{C, Ar, Ac, M, If\}$$

where:

*SDLC* refers to the software development life cycle being modeled.

*C* is a set of input and output criteria. These are discussed in section 4.4.

*S* refers to a set of stages within the software development life cycle.

*Ar* refers to a set of artifacts. Artefacts are discussed in section 4.3.

*P* refers to a set of processes. Processes are discussed in section 4.5.

*Ac* refers to a set of actors. Actors are discussed in section 4.2.

*M* refers to a process model. These are discussed in section 4.5.1.

*If* refers to a set of interface functions.

### 4.2 Actors

Actors are entities that participate in the development life cycle or have a stake in it. Due to the central role the actions of these stakeholders play in the success or failure of a project, they are a core element of the proposed language. An actor is defined using the *actor()* statement. One could define actors to be actual people or the roles the actors will be taking in a project. A few examples include *actor(Joe)*, *actor(Developer)* and *actor(Client)*. The following statements are defined for actors:

Statement	Description
<i>actor(x)</i>	Establishes <i>x</i> as an actor within the model.
<i>actorInProcess(x, p)</i>	Establishes <i>x</i> as a participant in process <i>p</i> .

The language also provides a number of functions which help to reason about the model. The following are a few examples:

Function	Description
$ActorInStage(x, s)$	<i>true</i> if actor $x$ participates in a stage $s$ .
$ActorInSDLC(x)$	<i>true</i> if actor $x$ participates in the SDLC.

$$ActorInStage(x, s) = true \Leftrightarrow \exists p \bullet (Process(p) \wedge ProcessInStage(p, s) \wedge ActorInProcess(x, p))$$

$$ActorInSDLC(x) = true \Leftrightarrow \exists s \bullet (Stage(s) \wedge ActorInStage(x, s))$$

### 4.3 Artefacts and Quality Attributes

The ultimate goal of a software development life cycle is to deliver a software solution and related artefacts such as deployment plans and training manuals. However, the life cycle also produces a number of intermediate artefacts which are used internally to produce the final outputted artefacts. Certain artefacts may also be provided as input to the life cycle from external sources. Artefacts exhibit quality attributes which when measured, can be helpful in the decision making process.

Statement	Description
$Artefact(x)$	Establishes $x$ as an artefact.
$QualityAttribute(x)$	Establishes $x$ as a quality attribute.
$InputArtefactToProcess(x, p)$	$x$ is an input artefact to process $p$ .
$InputArtefactToStage(x, s)$	$x$ is an input artefact to stage $s$ .
$OutputArtefactToProcess(x, p)$	$x$ is an output artefact of process $p$ .
$OutputArtefactToStage(x, s)$	$x$ is an output artefact of stage $s$ .

It is the intention of the authors to incorporate quality prediction capabilities into the language. As discussed in section 4.8, a vast amount of research has been done in this area and it would be useful to combine this with simulated scenarios so as to predict the quality of artefacts early on the the development life cycle.

### 4.4 Entry and Exit Criteria

A common way of controlling work flow in a development process is by having entry and exit criteria in place.

In our context, this means what conditions need to be met before a process, stage or even the whole life cycle can start or end. In most cases, criteria may be a simple “*the preceding process must terminate*” or “*the code must pass all unit tests with a minimum coverage of 80%*”). However, it is possible the have more complex scenarios. One may choose to include factors such as contractual obligations, quality of artefacts, financial situations and so on.

The language presented here provides a mechanism through which entry and exit criteria can be incorporated into the life cycle model. However, when it comes to expressing criteria, the choice was made to incorporate an existing language called the *Contract Language (CL)*. *CL* is a deontic language which was designed with the formalisation of electronic contracts in mind. It is beyond the scope of this paper to go into the details of *CL* but it suffices to say that *CL* allows one to specify obligation, prohibition and permission. In our context, criteria could say things like “*a process is must (is obliged) produce a requirements document*”, “*code cannot (is prohibited) be deployed if it does not pass all unit tests*” and “*one may ignore performance issues during prototyping the stage*”. This kind of expressive power is ideal for use in entry and exit criteria.

### 4.5 Processes

Processes are sub-components of stages and are meant to aid their parent stage in achieving its goals. In section 4.1, a process was formally defined as:

$$P := \{C, Ar, Ac, M, If\}$$

That is to say, a process is a collection of input and output criteria ( $C$ ), internal and external artefacts ( $Ar$ ), actors ( $Ac$ ), a process model ( $M$ ) and a set of interface functions ( $If$ ). Since criteria, artefacts, and actors have already been discussed in preceding sections, this section will focus on the process model and interface functions.

#### 4.5.1 Process Models and Interface Functions

A process model defines the actual inner workings of its process. The level of detail of this model can vary depending on one’s modeling requirements. The more detailed a model is, the more meaningful and accurate the analysis that can be carried out. Practitioners can build a process model using any language or notation which best suits the needs of the process being modeled. For example, if a process was a sequential set of steps, one might model it as a finite state machine or flowchart. On the other hand, if a process had concurrent aspects which needed modeling, one might choose to model it as a petrinet. The plugability

of these models into the language being proposed in this paper is achieved through the concept of *interface functions*.

Regardless of a process model's inner workings, there are a finite number of which one would ask of it. For example, one might be interested in whether a process is currently active or not, whether the process is on schedule, and so on. Similarly there is a finite set of control actions which one might want to assert over a process. Examples of these include suspending and resuming a process. The table below lists a number of example interface functions. Since it is impractical to list all interface functions in a conference paper, a representative sample of different types of interface functions has been selected.

Function	Description
<i>isActive()</i>	Returns true if the process is active.
<i>isComplete()</i>	Returns true if the process is complete.
<i>idleActors()</i>	Returns a list of actors who are currently idle within the process.
<i>statusDescription()</i>	Returns a human readable description of the process' status.
<i>unsatisfiedCriteria()</i>	Returns criteria which have not yet been satisfied.
<i>expectedCompletionTime()</i>	Returns the process' expected completion time.

Practitioners should implement these functions in second order logic and link them to their own process models. The authors are conscious that the difficulty of this may vary depending on the choice of modeling language for the process model. As a result, it is being proposed that bridging mechanisms be developed at a future stage which facilitate easier integration with common modeling language such as UML, petrinets and flow charts. This is discussed in section 4.8.

Interface functions are also expected to satisfy a number of conditions. These are meant to preserve the sanity of the model being built and server as means by which inconsistencies or any other problems could be flagged up. The following is a sample list of sanity conditions for interface functions.

$$isComplete(P) \rightarrow \sim isActive(P)$$

$$isActive(P) \rightarrow \sim isComplete(P)$$

$$isComplete(P) \rightarrow unsatisfiedCriteria(P) = \emptyset$$

## 4.6 Stages

A stage consists of a group of processes which work towards achieving a particular goal within the life cycle. A stage is mainly useful as a way of abstracting away from potentially complex interactions between processes so that it is easier for a life cycle to be modelled and understood. In section 4.1, a stage was defined as:

$$S := \{C, Ar, P\}$$

This definition simply establishes a state as being a collection of entry and exit criteria (*C*), a set of input and output artefacts (*Ar*), and a set of processes (*P*). The actual sequences in which processes are executed and how they communicate with each other will be jointly dictated by each individual process' entry and exit criteria which in turn may depend on input and output artefacts. All the concepts mentioned here have already been discussed in previous sections.

## 4.7 Querying and controlling the model

The discussion has thus far focused on how to actually define a model of a particular software development life cycle. Once the model is built, one would typically interact with it in three ways: *static queries*, *dynamic queries* and control instructions. This sections looks at each of these in turn.

### 4.7.1 Static Queries

Static querying involves querying the static structure of the model. This is usually done when one needs to understand how the model works. Typical examples in this class of queries include 'what stages does the life cycle contain?', 'what is a exit criteria of process P?', 'which actors are involved in stage S?', and so on. Since the foundation of the language is second order logic, a large number of these queries are provided automatically. However, for the sake of usability and convenience, a number of functions were defined so as to allow the user to avoid writing non-trivial logic queries. The following example illustrates a predefined function which allows the user to find out which actors participate in a particular stage.

$$actorsInStage(S) = A \bullet x \in A \Leftrightarrow isActor(x) \wedge \exists p \in processesInStage(S) \wedge isActorInProcess(x, p)$$

### 4.7.2 Dynamic Queries

Dynamic queries are not concerned with the actual structure of the life cycle but rather with the state of an instance



of the life cycle. Examples of dynamic queries include *'has artifact A been delivered?'*, *'what part of process P's exit criteria has not been met yet?'*, *'given the current circumstances, what is the predicted quality of artifact A?'*, and so on. If an instance of the model reflects real-life events, it can be used to monitor an actual running project. However, one could create scenario models which simulate events that might occur. Users can then query the simulation model to find out what would happen if those events occurred in real life. One can appreciate that this is a powerful concept which would improve the accuracy of, and confidence in decision making processes.

Although these queries differ from static queries, their actual implementation is similar. The difference is that instead of querying structural information, functions provided in this area query state information.

#### 4.7.3 Control Instructions

Control instructions are designed to modify the state of a model instance. A small number of these instructions are predefined but since they largely depend on the process models and process models can be defined by a variety of notations (see section 4.5.1), it is impossible to determine what functions will be needed in advance. As such, it is left up to the user to define control functions for custom-built models. Examples of control functions include actions such as marking an artefact as delivered, generating an even which drives a process whose model is represented by a finite state machine, suspending a process, and so on.

#### 4.8 Conclusions and Future Work

It is the believe of the authors that once expanded into a fully fledged language, the concepts discussed here will lead to a language which provides a powerful way of defining life cycle models and using them to monitor progress and simulate scenarios for problem solving and risk management. The work presented here is by now means complete and a number of issues still need to be addressed. One such issue involves the facilitation of defining process models in any language. This is a bold feature in the sense that it requires sophisticated mechanisms to link any arbitrary language into the one discussed in this paper. It is the authors' intention to fully specify these mechanisms and also utilise them to provide inbuilt support for popular languages such as UML, flow charts and petrinets.

The authors would also like to explore the area of software quality prediction. A substantial body of work has amassed in this area and it is the opinion of the authors

that it is useful to be able to predict quality of artefacts albeit to varying degrees of accuracy at any point during development.

The next few months will see ongoing work in this area as the authors flesh out the remaining syntax and semantics and encapsulate it in a language specification manual. This manual should also include examples of the language being used to model and simulate real-world life cycles. It is the ambition of the authors to be in a position to publish findings from such work by the end of the year.

#### References

- [1] Royce W., "Managing the Development of Large Software System", Proceedings of the IEEE Western Conference, 1970
- [2] Beck K., Extreme Programming Explained: Embrace Change, Addison-Wesley, 1999
- [3] Takeuchi H., Takeuchi I., The New New Product Development Game, Harvard Business Review, January 1986
- [4] PM Solutions, White Paper: "Selecting a software development life cycle methodology", 2003
- [5] Beck K., Fowler M., et al, "The Agile Manifesto", <http://agilemanifesto.org/principles.html>, 2001
- [6] Kiczales G., et al, "Aspect-Oriented Programming", Proceedings of the 11<sup>th</sup> European Conference on Object-Oriented Programming, 1997
- [7] Boehm B. W., "A Spiral Model of Software Development and Enhancement", IEEE Computer, pages 61-72, May 1988.
- [8] Smith J., Technical Report: "A Comparison of RUP and XP", Technical Report, IBM Library, 2003

# Weaving Process Patterns into Software Process Models

Xiao-yang He, Ya-sha Wang\*, Jin-gang Guo, Wu Zhou and Jia-kuan Ma  
Institute of Software, School of Electronics Engineering and Computer Science,  
Peking University, Beijing 100871, China;  
Key laboratory of High Confidence Software Technologies (Peking University),  
Ministry of Education, Beijing 100871, China

**Abstract** Defining an appropriate software process model is a knowledge-intensive, time-consuming and laborious work. However, practitioners have already accumulated abundant process knowledge in the form of solutions to recurrent problems within software development. Process pattern is a good way to capture and represent such knowledge. Most of current research leverages natural or semi-formal languages to describe process patterns. In order to apply the knowledge represented by those patterns, users have to understand the pattern knowledge in person, and then modify the process model manually, which is unefficient and error-prone. To solve this problem, a precise meta-model for process pattern is proposed, and an approach to weaving process patterns into software process models is provided.

## 1. Introduction

Defining an appropriate process model for software development project is knowledge-intensive. The process developer should have clear impression on project characteristics as well as abundant process knowledge. The knowledge here means general solutions to recurrent problems in software process. For facilitating reuse of such knowledge, process pattern is adopted to structurally represent and document the knowledge recently.

A process pattern describes a general solution to the development problem which occurs in different project environments [1]. Some approaches [1, 2] use natural language to describe process pattern. It may produce ambiguous and inexact expressions, and is hard to be processed by machine. Some approaches[3-5] describe patterns by workflow-graph-alike languages which are derived from traditional process modeling languages and just suitable for expressing complete, elaborated and detailed process models. They regard patterns as

black-boxes which can be combined together to construct the process model by interfaces. As a result, they could not make any adjustment of patterns to the specific project where patterns are applied. Therefore, the reusability of black-box pattern is very limited.

In our eyes, the essential elements of the pattern solution(such as activities, artifacts, roles, relationships, constraints, rules, etc.) always remains the same; however, when the pattern is adapted to specific requirements of each project, the general solution may take on different shapes in resulting processes. For successful reuse among different projects, intrinsic characteristics of the common solution should be separated from those of the project environments where pattern are applied. Based on this idea, a process pattern only specifies essential elements of the solution in this paper. It may describe a set of incomplete, loose process fragments, which will be supplemented by concrete process elements from the existing process.

Our approach consists of two parts: (1) representing a process pattern as a minimal set of essential elements; (2) weaving the pattern into existing processes automatically based on process structure analysis.

## 2. Process pattern representation

In our meta-model for process pattern (Fig.1), pattern solution is an *ExecutableNode* set to represent the essential activities and their temporal/logical relationships. Each *ExecutableNode* is either an *Activity* or a *StructuredActivity*. *Activity* is the work definition performed by certain *Role* which uses and produces some *WorkProduct*. A *StructuredActivity* groups its subordinate nodes (low-level *ExecutableNodes*) by their collaborative way. Classified by the collaborative ways of subordinate nodes, there are five kinds of *StructuredActivity*:

- **Sequence:** The subordinates are executed in order.
- **Parallel:** The subordinates are executed concurrently.
- **Choice:** The subordinates are chosen exclusively.
- **Cycle:** The subordinates are executed in order again and again until the stopping condition is satisfied.
- **And:** There is a loose temporal/logic relationship among

---

\* Contract Author: Ya-sha Wang (wangys@sei.pku.edu.cn)  
This work is funded by the National Basic Research Program of China (973) under Grant No. 2009CB320703, the High-Tech Research and Development Program of China under Grant No. 2007AA010301, and the Science Fund for Creative Research Groups of China under Grant No. 60821003.

the subordinates. The subordinates should have the same predecessor and successor. Apart from this, there is no other restriction among them. After applying the pattern to some existing process by way of weaving, *And* will be replaced by other determinate relationships (*Sequence*, *Parallel*, or *Cycle*) for reason of pattern adaptation.

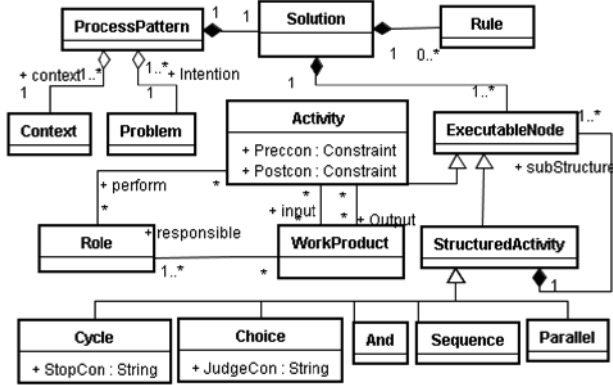


Figure 1 Meta-model for Process Pattern

The *ExecutableNode* set is presented by a tree structure called after Process Pattern Structure Tree (PPST) in this paper. Each leaf node in PPST is an *Activity*. Every sub-tree is a *StructureActivity* defining how its subordinate nodes are coordinated. The root node of the sub-tree is decided by the *StructuredActivity* type (*Sequence*, *Parallel*, *Choice*, *Cycle* or *And* which is represented with the symbol of  $\odot$ ,  $\parallel$ ,  $\ominus$ ,  $\odot$  or  $\wedge$  accordingly). The sub-trees do not refer to those only consisting of one leaf node. Every *Activity* belongs to a *SubstructureActivity* and is not shared with any other *SubstructureActivity*, except for nesting.

Fig. 2 is the PPST of Pattern “*Feasibility Analysis*” in [1], which means: (1) The pattern consists of six activities; (2) Activity “*Determine operational feasibility*”, “*Determine economic feasibility*”, “*Determine technical feasibility*” and “*Identify risks*” make up of a *StructuredActivity* whose type is  $\odot$  so their execution sequence is not appointed in the pattern; (3) Activity “*Identify alternatives*” should start to execute before the others and “*Choose an alternative*” should start to execute after the others.

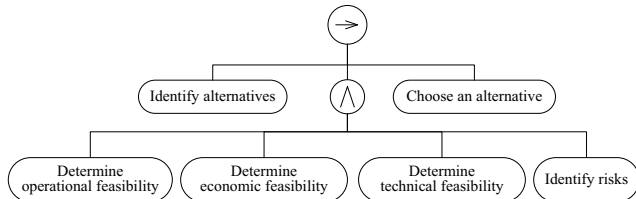


Figure 2 PPST of Pattern “Feasibility Analysis”

### 3. Process pattern application

The application of a process pattern to an existing process means merging two control flow structures and related activities into one. We provide an approach to

weaving process pattern with existing process automatically. Here, the existing process is called *source process* while the process after applying pattern is called *target process*.

In our approach, three important principles should be followed:

**Principle 1** It should be ensured that all constraints in the pattern are satisfied in target process.

**Principle 2** It should be ensured that the original behavior of source process is preserved in target process on condition of Principle 1.

**Principle 3** It should be ensured that unnecessary precedence relationships are not introduced in target process.

#### 3.1 Overview of the pattern application approach

The pattern application approach is clarified as follows.

**Phase 1: Parse the source process to a Process Structure Tree (PST).**

The control flow of a process model (for example, UML activity diagram, EPC, BPMN) can always be modeled as a workflow graph. Parse the workflow graph into a Process Structure Tree (PST)[6] is very useful to simplify the complexity of process analysis. The PST is so much alike PPST defined above except that PST does not have *And* relationship.

**Phase 2: Compare the pattern solution with the source process.**

Establish a mapping between the elements of the process pattern and those of the source process. To make a possible mapping, the process developer is responsible for determining similar behaviors from both namespaces. Without loss of generality, we assume that the “Behavior” namespaces are already synchronized.

**Phase 3: Weave the pattern solution into source process.**

Based on process structure analysis, the pattern solution is weaved into source process seamlessly. The activities and temporal/logical relationships in source process which are divergent from the pattern are rectified, and others information are kept untouched. In addition, the incomplete parts in the pattern may also be complemented by the pertinent information embodied in the source process.

**Phase 4: Simplify the PST.**

If necessary reformat the PST to eliminate redundant nodes and edges brought by application operations. For example, the node “parallel” has only one child node *A*, then the node “parallel” is removed from the PST and its father node is linked to *A* directly.

**Phase 5: Restore the PST to the target process.**

Preorder traverse the PST and convert it to a workflow graph of target process.

In our approach, Phase 1 can be implemented based on existing research and Phase 2 can be solved by synonyms or process ontology. It is easy to handle Phase 4 and 5 so that they are omitted for the space limitation. Phase 3 is the

focal point of this paper because of its importance and pivotal role.

### 3.2 Change modes

Weaving the pattern into existing process means certain application operations will be put on the process. Here, seven change patterns are discovered to solve the problem of how to change the process when pattern applied. For avoiding confusion, the change pattern is named as change mode.

In order to describe the change modes clearly, the variables to be used are defined in advance. Let  $PP$  be the PPST of the process pattern, and  $PS$  be the PST of the source process.  $MN$  denotes the set of nodes both in  $PS$  and  $PP$ .  $x, y$  and  $z$  are all node variables.  $MinComma(x, y, PS/PP)$  returns the minimal common ancestor of  $x$  and  $y$  in  $PS$  or  $PP$ .  $MinCommTree(x, y, z, PP/PS)$  is the  $PP/PS$ 's minimal sub-tree which  $x, y, z$  belongs to.  $IsBrother(x, y, PS/PP)$  is used to judge whether  $y$  is  $x$ 's brother node in  $PS/PP$  or not.

We also define the hierarchy and temporal relationship between the nodes in the tree. If node  $y$  is node  $x$ 's offspring, it is called  $y \ll x$ . If node  $y$  is node  $x$ 's right brother or in the sub-tree which root is  $x$ 's father right brother, and so on,  $y$  is called *After*  $x$ . Conversely,  $x$  is called *Before*  $y$ .

PS \ PP	⊖	⊙	//	⊕	⊗
⊖	K	R	R	/	K
⊙	K	R <sup>#</sup>	R <sup>#</sup>	/	K
//	R	R	K	/	K
⊕	/	/	/	R	/


Note:   $R^*$

Figure 3 Relation Conversion Table

By following **Principle 1** and **2**, *Relation Conversion Table* (RCT) (Fig. 3) is defined to determine the transformation of control dependency when pattern is applied. Suppose  $x, y \in MN$ . The row and column denote the parent of  $x$  and  $y$  in  $PP$  and  $PS$  respectively. The cross units represent the parent of  $x$  and  $y$  in the target process after applying pattern. 'K' means that node in  $PS$  are kept while 'R' means that node in  $PS$  will be replaced by the node in  $PP$ .  $R^{\#}$  is a special situation that the stop condition of the node in target process should get the intersection set from that of  $PP$  and  $PS$ . '/' denotes an impossible situation.

#### Change Mode 1 Disposition in couples

**Problem** How to change the control flow in  $PS$ ?

**Context**  $x, y \in MN$ , and  $IsBrother(x, y, PP)$  is true, and there does not exist  $z (z \in MN, IsBrother(y, z, PP)$  is true, and  $MinComma(x, z, PS) \ll MinComma(x, y, PS)$ ).

**Solution** Dispose the pair of  $MinComma(x, y, PS)$  and  $MinComma(x, y, PP)$  according to RCT.

#### Change Mode 2 Order transposition

**Problem** How to change the control flow in  $PS$ ?

**Context**  $x, y \in MN$ , and  $IsRBrother(x, y, PP)$  is true, and

$y$  is *Before*  $x$  in  $PS$ , and  $MinComma(x, y, PP) = \ominus$  or  $\odot$ , and (there does not exist node  $z, z \in MN$  and  $z$  is *After*  $x$  and *Before*  $y$ ).

**Solution** Move  $y$  to the place of  $x$ 's left brother.

#### Change Mode 3 Monolithic permutation

**Problem** How to change the control flow in  $PS$ ?

**Context**  $x, y, z \in MN$ ,  $IsBrother(x, y, PP)$  is true and  $IsBrother(x, z, PP)$  is false, and  $MinComma(x, z, PS) \ll MinComma(x, y, PS)$ ).

**Solution** Check the relation of each pair among  $x, y$  and  $z$  in RCT. Keep  $PS$  untouched if each return value is 'K', or else replace  $MinCommTree(x, y, z, PS)$  by  $MinCommTree(x, y, z, PP)$  monolithically.

#### Change Mode 4 Parsing group

**Problem** How to change the control flow in  $PS$ ?

**Context**  $x, y, z \in MN$ , and  $IsBrother(x, y, PP/PS)$  is true and  $IsBrother(y, z, PP/PS)$  is true.

**Solution** Lower the depth of  $x, y$  in  $PP/PS$  and create a new node as the father node of  $x$  and  $y$ . Then dispose them like Pattern 1, 2 or 3.

#### Change Mode 5 Single node permutation

**Problem** How to change the control flow in  $PS$ ?

**Context**  $x \in MN$  and there does not exist node  $z (z \in MN, IsBrother(x, z, PP)$  is false and  $IsBrother(z, x, PP)$  is false).

**Solution** Compare  $x$ 's father nodes both in  $PP$  and  $PS$ . If they are not same and  $x$ .parent =  $\ominus$  or  $\odot$  in  $PP$ , insert  $x$ 's father node in  $PP$  into  $PS$ , or else keep  $PS$  untouched.

#### Change Mode 6 Add an activity

**Problem** How to insert an activity in  $PS$ ?

**Context**  $z$  is a leaf node,  $z \in PP$  and  $z \notin PS$ .

**Solution** Find  $z$ 's left brother node  $x$  (if  $z$  has no left brother,  $x$  is left brother node of  $z$ ' father, and so on) and right brother node  $y$  (if  $z$  has no right brother,  $y$  is right brother node of  $z$ ' father, and so on).  $z$  may be inserted into any position in the path from  $x$  to  $y$  in  $PS$ . Choose the best candidate by **Principle 3**.

#### Change Mode 7 Delete an activity

**Problem** How to delete an activity in  $PS$ ?

**Context**  $PP$  has a rule "NOT Require Activity  $x$ ".

**Solution** Remove  $x$  if it exists in  $PS$ .

A weaving algorithm based on process structure analysis is proposed to facilitate merging the pattern into the process. The compliance of source process with pattern is checked, and then the corresponding changes modes are applied to those incompliant parts in source PST. Because of space limit, the detail of weaving algorithm could not be shown in this paper.

## 4. Illustrating example

One application of our approach is to ensure a qualified process. Process pattern can be used to define the necessary constraints to ensure a process with high quality, and then

verify whether all the constraints in the pattern are satisfied in existing process. If not, those unsatisfied parts will be added, deleted or corrected.

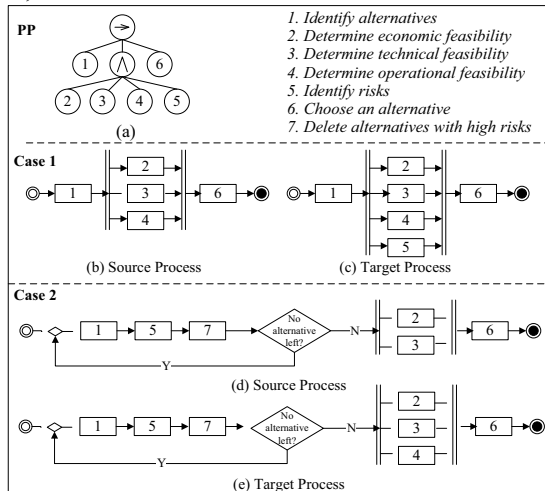


Figure 4 Example of “Feasibility Analysis”

Pattern “Feasibility Analysis” has already been described in Section 2. Now it is applied to two cases. In Case 1(Fig. 4(b)), the alternatives are identified firstly, and then economic feasibility, technical feasibility and operational feasibility are determined parallel to choose an alternative. Case 2 (Fig. 4(d)) is a little different from Case 1. After identifying alternatives, risks of each alternative are evaluated and those with high risks are discarded. If there is still any alternative left, make a choice after assessing the economic and technical feasibility. Compared with the pattern, Case 1 does not have the activity of identifying risks while Case 2 lacks the activity of identifying operational feasibility. The resulting processes of Case 1 and 2 are shown in Fig. 4(c) and Fig. 4(e) respectively, and they demonstrate totally different shapes. The activity “Identify risks” is parallel to “Determine economic feasibility” in Case 1 but before “Determine economic feasibility” in Case 2. Moreover, Case 2 keeps the activity “Delete alternatives with risks” and its logic relationships which are not occurred in the pattern.

From the example, it is can be seen that the pattern described by our approach is capable of providing high reusability and feasibility. Since the pattern only defines the essential activities and relationships, the applied process can keep itself activities and relationships as long as it does not violate pattern’s constraints.

## 5. Related work

The expected interests of reduced development time and improved process quality make pattern approach attractive. However, this concept still remains difficult to be exploited in practice due to the lack of formalization and supporting methodology [4].

Some research regards process pattern as a black box

which may combined together to construct a process, such as [3-5]. They do not break away from the concept of process component; consequently, they can only provide small-grained reuse. All of these process pattern descriptions are short of flexibility and hard to incorporate into tool support. Tran etc. [7] broaden the use of process patterns and begin to concern with the pattern's effect on process structure, but their approach can only be done manually. Foster[8] introduces a method to represent the variabilities in the pattern, but he still does not solve the problem of merging pattern with existing process either.

The problem of parsing workflow graph is deeply discussed in business process community. A parsing algorithm is proposed in [6] to ensure obtaining a unique, modular parsing, which lays the foundation of our work.

## 6. Conclusions and future work

Process pattern is a useful means of capturing and reusing process knowledge. Our approach only describes everything that is elementary for the pattern so it may enable high flexibility when pattern is applied. We also provide an application approach to automatically weaving a pattern into processes. In particular, the notion of process structure tree is adopted to simplify the complexity of process analysis.

We are now developing the supporting tool for pattern representation and automatic weaving. Further work is also concerned with how to identify the correspondences between process pattern and source process automatically.

## 7. Reference

- 1.Ambler, S.W., *Process Patterns: Building Large-Scale Systems Using Object Technology*. 1998: Cambridge University Press.
- 2.Coplien, J.O., *A Generative Development - Process Pattern Language*, in *The Patterns Handbook: Techniques, Strategies, and Applications*.1996, Cambridge University Press. p. 243-300.
- 3.Gnatz, M. *Modular Process Patterns Supporting an Evolutionary Software Development Process*. in *PROFES 2001*. 2001.326-340
- 4.Hagen, M., V. Gruhn. *Towards flexible software processes by using process patterns*. in *Software Engineering and Applications*. 2004.436-441
- 5.Störrle, H., *Describing Process Patterns with UML* Software Process Technology, 2001. LNCS 2077:173-181.
- 6.Vanhatalo, J., H. Völze, J. Koehler. *The Refined Process Structure Tree*. in *BPM 2008*. 2008: Springer-Verlag Berlin Heidelberg
- 7.Tran, H.N., B. Coulette, B.T. Dong. *Modeling Process Patterns and Their Application*. in *International Conference on Software Engineering Advances*. 2007
- 8.Förster, A., G. Engels, T. Schattkowsky, et al. *A Pattern-driven Development Process for Quality Standard-conforming Business Process Models*. in *VL/HCC 2006*. 2006.135 - 142

# Assessing Workflow Ability of ERP and WfM Systems

Lerina Aversano, Roberto Intonti and Maria Tortorella  
(aversano/intonti/tortorella@unisannio.it)

Department of Engineering, University of Sannio  
Via Traiano, Palazzo ex-Poste – 82100, Benevento, Italy

***Abstract** - Automation of a business process can be obtained by using a workflow management system or an ERP system embedding workflow functionalities. The wider diffusion of ERP systems tends to favorite this solution, but there are several limitations of most ERP systems for automating business processes. Actually, there is a lack of empirical studies aiming at achieving an evidence of these limitations. The work proposed in this paper goes in this direction. It reports an empirical study aiming at assessing the “workflow ability” of ERP systems and comparing it with that of Workflow Management Systems*

## 1. INTRODUCTION

Fast changing business requirements forces enterprises to support their business processes with appropriate software applications in order to effectively execute the related business activities. The software systems mainly considered from enterprises with this purpose belong to the following two categories: Workflow Management Systems (WfMS) and Enterprise Resource Planning (ERP) systems.

The two different solutions deal with the processes by using different approaches. ERP systems essentially represent multi-module applications integrating activities across functional departments, from product planning, purchasing, inventory control, product distribution, to order tracking [14, 15]. They are designed around the idea of applications that need to be configured with appropriate setting for achieving the solution most adequate to the enterprise requirements. Of course, higher the possibility for configuration is, more flexible the support for the business process is. On the other side, WfMSs are a new kind of information technology designed for automating business processes by coordinating and controlling the flow of work and information between participants.

When a WfMS is used, a workflow model is first defined for the specific requirements of the enterprise and, then, workflow instances are created for performing the actual activities described in the workflow model.

The paper reports an empirical study aiming at assessing the “workflow ability” of ERP systems and comparing it with that of Workflow Management Systems.

The paper is organized as follow: Section 2 gives some background on the study and presents related work; Section 3 describes what is meant for workflow ability and introduces the approach used for the empirical study; Section 4 describes the used systems and the main results achieved; finally, conclusions are discussed in Section 5.

## 2. THEORETICAL BACKGROUND AND RELATED WORK

Workflow Management System is a technology mainly focused on the automation of business processes. It is widely adopted to support production activities of people in enterprises [1, 2]. ERP systems, viceversa, mainly address the need of having an integrated database that serves different functional modules. In the literature, there are several definitions of ERP systems [4, 5, 13].

Basically, ERP systems overcome the data separation of multiple functional applications by allowing individual modules to share the same data. Moreover, most of them contain functionality to model, deploy and manage workflows. The ERP “embedded” Workflow System is a module which is a part of the core ERP architecture.

Works related to the study presented in this paper mainly focus on the differences existing between ERP systems and WfMSs and/or definition of frameworks and approaches for facilitating their integration. A strategy proposed for their integration consists in the use of a WfMS as a mean for implementing a workflow controlling the ERP functionalities [6]. The problem of this approach was highlighted in [5] and mainly deal with the difficulties of managing inconsistencies between the two systems.

Several other strategies address the problem of the integration by considering the WfMS as a “middleware” orchestrating legacy applications and ERP systems. Newmann and Hansmann [7] developed an architecture for integrating WfMS and the planning functionality embedded in ERP systems. Brehm and Gomez proposed

an approach for federating ERP systems exploiting an architecture based on a WfMS enacting Web Services implementing the ERP functionalities [8]. Tarantilis, Kiranoudi and Theodorakopoulos presented an ERP web-based system allowing the management of supply chain embedding a workflow engine for managing the control flow of the enterprise [9]. Szirbik and Wortmann proposed an agent based architecture for integrating ERP and WfMS. The agent was a bridge able to access relevant global information from the ERP transaction database and reason about the situation in the workflow [10]. Namin and Shen proposed a Web services agent based model for inter-enterprise collaboration [11].

### 3. EVALUATION CONTEXT

In this paper, Workflow ability is intended as the capacity of a software system of effectively and efficiently supporting the modelling, execution and monitoring of a business process. It has to include also the easy interaction with external software systems for executing the activities. On the basis of this definition, an evaluation framework has been defined with a set of Workflow ability characteristics to measure in both ERP and WfM systems.

#### 3.1. EVALUATION FRAMEWORK

The framework is based on four parameters: *Process Modelling Complexity*, which represents the capability of the system of modelling complex business processes. *Modelling Tool Usability*, considering how easily the software system permits to model business processes. *Process Execution Support*, evaluating the capability of the software system of managing the business process after its modelling and during its execution. *Software System Interoperability*, which analyses how easily the software system interacts with the software systems supporting business process activities.

High values of the parameters indicate that a system owns high workflow ability. Using the GQM paradigm, the following four goals were defined:

**GOAL 1:** Analyze an ERP/WfMS software system with the aim of evaluating its Process Modelling Complexity Level from the point of view of the business process analyst.

**GOAL 2:** Analyze an ERP/WfMS software system with the aim of evaluating its Modelling Tool Usability Level from the point of view of the business process analyst.

**GOAL 3:** Analyze an ERP/WfMS software system with the aim of evaluating its Process Execution Support from the point of view of the software engineer.

**GOAL 4:** Analyze an ERP/WfMS software system with the aim of evaluating its Software System Interoperability from the point of view of the software engineer.

Id	Questions	Description
<b>Goal 1: Process Modelling Complexity</b>		
Q1.1	Which kind of languages is used for process modelling?	The language can be proprietary, object relation model, Script, XML.
Q1.2	Which dept level can be considered for process models?	A process can be expressed through sub-processes, sub-activities and tasks.
Q1.3	Which is the completeness level for defining a business process?	Each business process component can be described indicating data flow, allocated resources, relations with other components.
Q1.4	Is it possible to validate a business process before its execution?	Validation tools are usually available in advanced systems specific to WfM
Q1.5	Is it possible to design the Enterprise Organization?	Organization design tools usually available in advanced systems
Q1.6	Which type of control structure does the software system manage?	A business process can include cyclic, concurrent, complex, splitting and joining activities
<b>Goal 2: Modelling Tool Usability</b>		
Q2.1	In which format are displayed the process to the user?	The process can be shown in graphical, formal, semi-formal, textual format
Q2.2	How can the definition and/or modification of the business process executed?	A process can be defined and/or modified through a graphical and textual editor, or its formal representation can be interpreted
Q2.3	Do the system include default predefined process models?	A predefined process can exist described at process, activities and/or task level
Q2.4	How easily understandable is the software system interface?	Terms used for indicating the available functionalities, their grouping and the existence of help supports are investigated at the interface level
Q2.5	How easy is to perform the definition of a business process ?	The functionality for defining a process, its activities and tasks, and assigned resources should be easy to be performed
Q2.6	How easy is to remember how to use the system?	The needed functionalities should be easily identifiable and their nesting dept should not be high
<b>Goal 3: Process Execution Support</b>		
Q3.1	Which functionalities are available for the workflow execution at process level?	Definition, activation, suspension, and termination functionalities available for managing the execution of processes are identified and evaluated
Q3.2	Which functionalities are available for the workflow execution at activity level?	Definition, activation, visualization, and termination functionalities available for managing the execution of activities are identified and evaluated
Q3.3	Which functionalities are available for the workflow execution at task level?	Definition, activation, visualization, and termination functionalities available for managing the execution of tasks are identified and evaluated
Q3.4	Does the system permit to monitor the business process execution?	Monitoring services such as production of reports regarding state of an executed process activities, quality attribute of a process are evaluated
Q3.5	How are acquired the simulation information?	Process simulation tools should be included for managing processes
Q3.6	Does the system manage the alerts?	Mechanism alerting for unexpected execution should be considered
Q3.7	How flexible is the system for modifying a business process?	A business process can need to be dynamically changed with its resources even at run-time
<b>Goal 4: Software System Interoperability</b>		
Q4.1	Which import and export formats for data exchange does the system support?	The format (XML, PDF, MSOff, ecc ...) for data exchange provided by the systems are analysed
Q4.2	In which language the process has to be formally described for being exported?	The format (XML, BPEL, Defjdl, XPDL, ecc ...) for exporting model process are investigated
Q4.3	Which databases are supported by the system?	The databases the software system supports are listed and evaluated
Q4.4	Does the software system support the use of Web Services?	The existence of a support of Web services for managing the process activities is detected

Table 1 - Workflow ability Evaluation framework

The achievement of each goal requires its characterization and identification of the aspects to be considered and evaluated. In conformance with the GQM paradigm, the characterization is obtained through the formulation of a set of questions aimed at understanding the level each parameter could reach. Table 1 reports the characterization of the goals in terms of the set of the investigated questions. The third column of the table contains a short description of each question with an indication of the investigated metrics. For brevity, just the investigated parameter is indicated for each goal.

QUESTIONS AND METRICS	
<b>Goal 1</b>	
<b>Q1.6</b>	<b>Which type of control structure does the software system manage?</b>
<i>M1.15</i>	Management of business processes with cyclic sub-processes or activities
<i>M1.16</i>	Management of business processes with splitting and joining sub-processes
<i>M1.17</i>	Management of business processes with concurrent activities
<i>M1.18</i>	Management of business processes including with arbitrary complex rules
<b>Goal 2</b>	
<b>Q3.6</b>	<b>Does the system manage the alerts?</b>
<i>M3.18</i>	Generation of alerts for overcoming the predefined end time of a task
<i>M3.19</i>	Generation of alerts for overcoming the predefined end time of a process
<i>M3.19</i>	Generation of alerts for indicating that there are task without users assigned
<i>M3.20</i>	Generation of alerts for indicating that there are tasks with too many users assigned
...	

Table 2 - Fragment of the evaluation framework

Table 2 contains an example of the defined measurement framework. In the interest of brevity, it includes just two fragments: one refers to the *Business process models control structuring*, chosen for answering Goal 1; and the other one refers to the *Business process execution monitoring* needed for evaluating the Goal 3.

Almost all the parameters are evaluated in a scale ranging from 0 to 4. Only the parameters of Goal 2 are evaluated in a scale ranging from 0 to 3. Answering each question considers an average of the parameters values. In the same way, evaluating the goals considers an average of the results obtained for each question.

### 3.2. ASSESSED SYSTEMS

Six software systems were evaluated: three ERP systems and three WfMSs. They are the following:

1. **Openbravo ERP** (<http://www.openbravo.com/>): a Web based ERP for SME, built on proven MVC & MDD framework that facilitate its customization.
2. **Compiere ERP/CRM** (<http://www.compiere.com/>): an open source ERP and CRM business solution for the

Small and Medium-sized Enterprise (SME) in distribution, retail, service and manufacturing.

3. **OpenERP** (<http://openerp.com/product.html>): an Open Source enterprise management software. It covers and integrates enterprise needs and processes: accounting, sales, CRM, purchase, stock, production, services and project management, marketing, and so on.
4. **Ultimus BPM Suite** (<http://www.ultimus.com/>): a solution for automating essential business processes. It handles order processes, purchase requisitions, claims processes, document reviews, and so on.
5. **JBoss jBPM** (<http://www.jboss.com/products/jBPM>): a system for creating business processes coordinating people, applications, and services. It brings process automation to business problems from embedded workflow to enterprise business process orchestration.
6. **ProcessMaker** (<http://www.processmaker.com/>): an Open Source business process management and workflow software including It includes tools to enable management of operational processes across systems including finance, human resources and operations.

## 4. EVALUATION RESULTS

The next four subsections discuss the results obtained for the four goals.

### 4.1. PROCESS MODELLING COMPLEXITY

Figure 1 shows that the WfMS permits to achieve a higher level of complexity in the Business Process Models parameter. In particular, Ultimus is the WfMS more efficient from this point of view, while OpenBravo and Open ERP obtain better values among the ERP systems.

The table in the Figure 1 evicts that the ERP systems obtained the worst results compared with the WfM systems as they lack any support for defining complex rules, mechanisms of split/join processes (with the exception of OpenERP) and task management.

Similar results were reached with reference to the business process structure complexity, Q1.2. In fact, only two WfM (JBoss JBPM and Ultimus) systems support the process decomposition in sub-processes, activities and tasks. The other ERP systems do not support such a kind of decomposition, while Process Maker does not include the task management without a prior implementation effort. With reference to Q1.3, it is possible to notice that only the WfMSs allow to specify details regarding inputs/outputs of activities/tasks. Finally in Q1.4, it is possible to observe that only Compiere includes the validation functionality. While, just Ultimus permits the designing of the organization, Q1.5..

The better results obtained for the WfMS are justified by the fact that the main scope of WfMS is the automation of business processes. Then, they were specifically



designed for flexibly capturing and representing all the process characteristics.

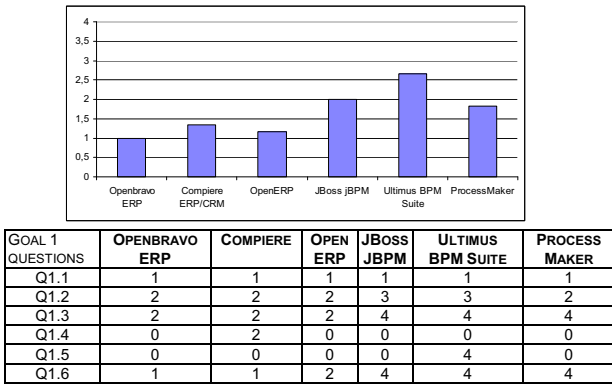


Figure 1 – Results per questions regarding the Process Modelling Complexity

#### 4.2. MODELLING TOOL USABILITY

Figure 2 shows that regarding the modelling tool usability, both ERP and WfM systems provide the same performance from a general point of view. The main differences can be evicted when the analytical data are analysed. In particular, the table in Figure 2 highlights that the main differences regard the availability of predefined templates and use of the user interface. More precisely, with reference to Q2.3, predefined process models are more available in ERP systems than WfMSs that, on their side, include more flexible definition tools, in fact they exhibit easier process model definition (Q2.5) and functionality remembering (Q2.6).

Regarding the other questions, it can be evicted that the achieved values are equal for the ERP systems, highlighting their similarity from the point of view of the analysed characteristics. In particular, regarding Q2.1, all the analysed systems consider similar format for representing business processes. JBoss JBPM makes an exception, as it uses different mechanisms. An similar evaluation is reached for ERP and WfM systems with reference to Q2.2. The differences are evicted by analysing the systems implementation. In particular, the ERP systems perform the definition and modification of business processes through the filling of forms were activities, tasks and resources are specified. OpenERP provides also a graphical modeller, but it is very complex to use. WfMSs allow the definition and modification of business processes by using a graphical modeller that permits to compose a business process through drag and drop operations. Finally, regarding Q2.4, the ERP systems appears to be more user-friendly than WfMSs.

Also these data confirm that the characteristics closer to the process modelling activities are better observed in WfMSs. On the contrary, ERP systems encourage the use

of predefined solutions. For reaching richer and more flexible business processes, WfMSs may require expert users and implementation activities.

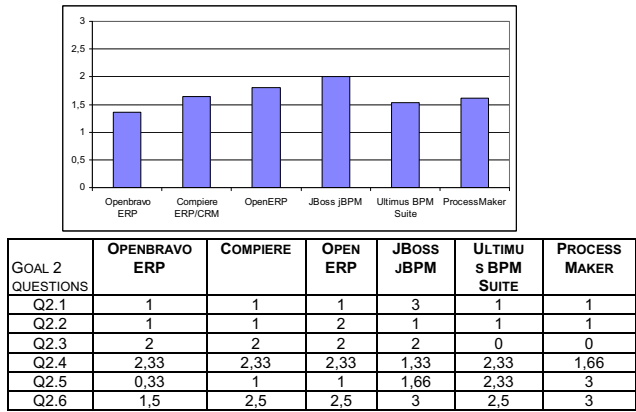


Figure 2 – Results per questions regarding the Modelling Tool Usability

#### 4.3. PROCESS EXECUTION SUPPORT

Figure 3 shows results achieved for the process execution support. It is possible to note that overall the WfMSs have better results. This is mainly due to the poor support for managing multiple instances of a process (Q3.1 and Q3.2). In detail, among the ERP systems with the best performance are OpenBravo (for its flexible management of alert and report) and Compiere (for its flexible management of alert and simulation).

Ultimus BPM Suite achieved the better result among the WfMSs, because it is the only system providing facilities for process simulation (Q3.5) and provides functionality for flexibly obtaining and customizing process execution report (Q3.4).

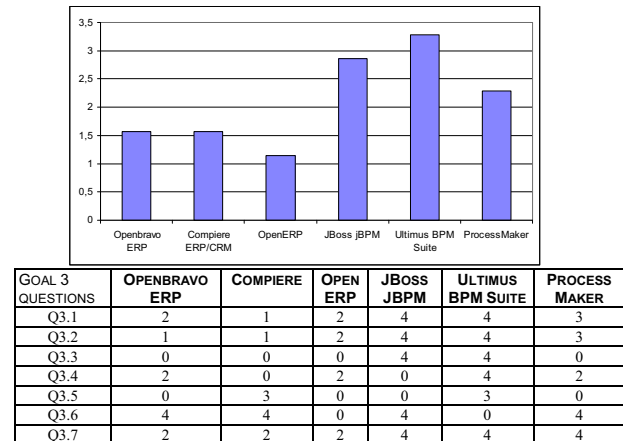


Figure 3 – Results per questions regarding the process execution support

With regards to the alert management (Q3.6), systems that provide the user with a flexible management of alert are Openbravo, Process Maker and Compiere. Alert

management with JBoss JBPM requires a significant programming effort for the user.

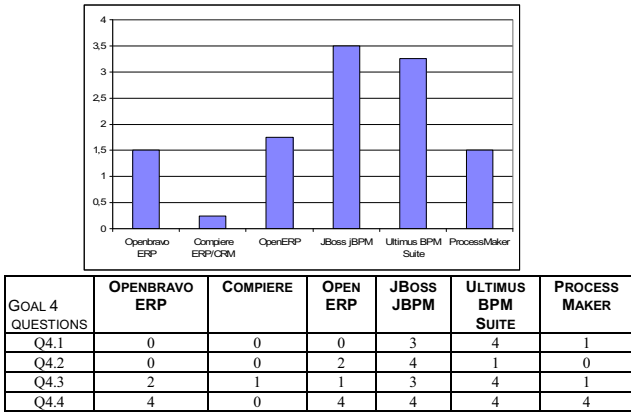


Figure 4 – Results per questions regarding the software system interoperability

#### 4.4. SOFTWARE SYSTEM INTEROPERABILITY

Figure 4 shows the results obtained for the software system interoperability. From the figure emerges that JBoss JBPM is the workflow system with an higher value of interoperability. Indeed, it supports different languages for the export of the process model and, in addition, interacts with multiple databases (Q4.3).

The systems having the lower value for the interoperability are Compiere, mainly due to the poor support for web service technology, and ProcessMaker, due to the use of proprietary language for process model.

#### 5. CONCLUSIONS

Both WfMSs and ERP systems play a major role in the automation of enterprise’s process. However, some research makes a comparison among these systems and concludes that ERP systems present some limitations for business process management. This conclusion does not have any quantitative evidence.

This paper presents a framework for assessing the workflow ability of both WfMSs and ERP systems. Results of the study confirmed the formulated hypothesis and WfMSs highlighted a major workflow ability for Process Modelling Complexity, Process Execution Support and Software System Interoperability. Similar results between the two kind of systems were obtained for Modelling Tool Usability, even if ERP systems obtained better values in those aspects regarding the peculiar way they manage the business processes that does not contribute to improve the workflow ability.

In the future, the authors will investigate the execution of further experiments concerning the workflow

ability evaluation in a larger set of systems, even analysing additional parameters.

#### REFERENCES

- [1] David Hollingsworth, “Workflow Management Coalition – The Workflow Reference Model”, Document Number TC00-1003, 19 Gennaio 1995: <http://www.wfmc.org/standards/docs/tc003v11.pdf>.
- [2] Layna Fischer, “Workflow Handbook”, Future Strategies Inc., 2002,
- [3] Hans Wortmann, Nick Szirbick, “ERP and Workflow Systems – Do they work together?”
- [4] Alshawi, Sarmand; Themistocleous, Marinos; Almadani, Rashid, “Integrating diverse ERP systems: a case study”, The Journal of Enterprise Information Management Volume 17, Number 6, 2004, pp.454-462, <http://www.emeraldlibrary.com/Insight/viewContentItem.do?contentType=Article&contentId=1529220>.
- [5] Liaquat Hossain, Jon David Patrick, M.A. Rashid, “Enterprise Resource Planning: Global Opportunities & Challenges”, pubblicato da Idea Group Publishing, 2002, pp. 16-17;
- [6] Jorge Cardoso, Robert P. Bostrom, Amit Sheth, “Workflow Management Systems and ERP Systems: Differences, Commonalities, and Applications”, <http://www.terry.uga.edu/~ekarah/cardoso.pdf>.
- [7] Stefan Neumann, Holger Hansmann, “WORKFLOW-INTEGRATED ERP: AN ARCHITECTURE MODEL FOR OPTIMIZED COORDINATION OF INTRA AND INTERORGANIZATIONAL PRODUCTION PLANNING AND CONTROL”, ECIS 2002 • June 6–8, Gdańsk, Poland.
- [8] Nico Brehm, Jorge Marx Gómez, “Service-oriented Development of Federated ERP Systems”, Proceedings of the SEMSOA Workshop 2007 on Software Engineering Methods for Service-Oriented Architecture, Hannover, Germany, May 10-11, 2007.
- [9] C.D. Tarantilis, C.T. Kiranoudis, N.D. Theodorakopoulos, “A Web-based ERP system for business services and supply chain management: Application to real-world process scheduling”, European Journal of Operational Research, 2008, vol. 187, issue 3, pages 1310-1326
- [10] N.B. Szirbik and J.C. Wortmann, “Bridging the gap between ERP and WFM in Planning using Agents”, Proc. of the Intl. IMS Forum 2004, Como, Italy, 17-19 May,
- [11] Akbar Siami Namin and Weiming Shen, “WEB SERVICES 1 AGENT-BASED MODEL FOR INTER-ENTERPRISE COLLABORATION”, IFIP International Federation for Information Processing, Springer Boston, Volume 159/2005.
- [12] APICS (2001). American Production and Inventory Control Society (APICS), <http://www.apics.org>.
- [13] Davenport, T. H., “Putting the enterprise into the enterprise system. Harvard Business Review”, 76( 4), 121-131.
- [14] Kumar, K. and Van Hillsgersberg, J, “ERP experiences and evolution. Communications of the ACM”, 43(4), 23-26.
- [15] O’Leary, “Enterprise Resource Planning Systems : Systems, Life Cycle, Electronic Commerce, and Risk. UK: Cambridge University Press”.

# Mining Objective Process Metrics from Repository Data

Michael VanHilst, Shihong Huang  
Florida Atlantic University  
777 Glades Road  
Boca Raton, Florida  
(mike, shihong)@cse.fau.edu

## Abstract

*The configuration management repository includes abundant data not only on configuration items, but about the process itself. But meaningful information about the software process is often hidden. This paper presents a method of extracting software process metrics from software repositories. More specifically, the metrics presented use data from the bug or task tracker and from the configuration management event log. The metrics are presented in graphic forms common to traditional and lean project management practices. The metrics presented here are empirical – not subject to bias in reporting or interpretation, and focused on measuring the process itself - not the developers and artifacts. They are derived from data that commonly exist in project software repositories, and thus can be collected with little or no cost. The metrics are illustrated with real software development repository data collected from a large industry project over a time span of several years.*

**Keywords:** software process improvement, process metrics, information visualization, data analysis, and software repository

## I. INTRODUCTION

Today, many software organizations make disciplined use of artifact management tools. These tools and practices create a unique source of development metrics. The metrics are objective – tied to actual process events, and non-invasive – extracted without additional developer input or effort [1]. The likelihood that any organization that is interested in metrics already uses tools and disciplined practices is high. In this paper we present a collection of software process metrics using only the event logs from two common development tools: bug/task tracking and configuration management.

This paper focuses on the collection and visualization of metrics that characterize the process itself. Our original interest was motivated by recent literature on process improvement. Common notions of process improvement address uncertainty and failure, and focus on conformance to best practices and improving estimates of time and effort. This earlier work gives us a range of

metrics to measure the complexity of code and the behavior of developers. More recent literature on process improvement, has been motivated by Agile and Lean practices and the application of Six Sigma to software. It shifts the focus to reducing cost and improving time to market. Improvement requires changes to the process. The measurements we present here support such change with baseline and comparison metrics. The metrics we use for progress, throughput time, and effort are not new. But the approach of deriving them non-invasively and at low cost from repository event logs is new.

We validated our approach on data collected from several large multi-month commercial projects involving hundreds of developers, thousands of artifacts, and tens of thousands of events. We illustrate the examples in this paper with graphs produced from one of those project.

Section II presents motivation and related work, Section III explains how the data is extracted from the repository and regrouped for analysis. Section IV presents several different graphical views of the process. Section V concludes with a brief comment on future work.

## II. MOTIVATION AND RELATED WORK

Often metrics that are described as ‘process metrics’ actually measure individual developers or properties of specific artifacts [2]. This is especially the case where software process improvement is viewed as improving conformance to existing practice. By measuring only how well developers are working, regardless of the problem being defined, the solution is for developers to work harder or with greater care. Our developers are already working overtime. We wanted to complement the data with measurements of how the process itself is working, independent of developer behavior.

We feel that with today’s competing development methodologies, there is more need for more metrics, and a greater variety of metrics, than ever before. Boehm and Turner, for example, list traditional engineering measurements as a significant barrier to implementing agile processes [3]. Their conclusion identifies the need for metrics data to validate value and capture lessons learned as “most important.” But no specifics are given.

[4][5] and others have surveyed the literature on software process improvement. The emphasis seems always to be on process adoption with little mention of improving business performance. [4] goes so far as to say that the current problem is a lack of effective strategies to implement process standards and models.

Process metrics that do exist often use questionnaires. The data is necessarily subjective and costly to collect. Overheads range from tens of person hours for CMMI Class C or ISO 15504 appraisals, to hundreds of person hours for a full CMMI Class A appraisal [6][7]. A supposedly light weight software process assessment describe in [8], recently spent 979 and 1376 hours in interviews.

[9] presents measurement models that are perhaps the closest to those presented here. But the ‘repository’ data is actually from manual reporting tools and is used not to assess the process, but how well individuals conform to it.

Hartmann and Dymond [10] provide guidelines and criteria for metrics emphasizing business value for use with agile and lean methodologies. But no actual metrics are described. The metrics presented here are consistent with those criteria.

The Poppendiecks [11] include measurement models, such as value stream maps and pareto charts, in their discussion of lean software development. But they don’t discuss how to collect the data. Their mention of Little’s Law, to estimate queue waiting and completion times, inspired us to investigate its use further. Little’s Law is described with more detail in [12].

Johnson et al. describe collecting data automatically from sensors in various development tools [13]. While this work complements our own (i.e. Johnson’s daily build metrics). But, like the others, they focus either on efficiency as a characteristic of individuals or improving the accuracy of predictions within a given process. Work with repositories often use data from open source project. Results from such data may not apply to industry practices. Fix times in [14], for example, were 200 days. Fix times in our data are typically less than 5 days.

Earlier tools designed to collect data from integrated development environments, such as TAME [15], AMADEUS [16], and APSE [17], assumed the existence of a process model. They would then measure and analyze how well the process plan was being executed. In practice, this approach yielded little value and eventually died. We don’t assume a process model. The model is revealed in the data. Ultimately, we are not as interested measuring plan execution as assessing process design.

### III. DATA COLLECTION AND REDUCTION

A software development project is made up of work pieces called tasks. Tasks divide a large project into small manageable units of work. Tasks are assigned to specific

developers, they trace specific work to a specific deliverable, and their completion represents tangible progress. A task can represent a changed or added piece of functionality, or a defect to repair. While useful as a management unit, different tasks can differ in both effort consumed and value produced.

A task begins with an initial request and ends with final integration into the product. As they make their way through the development process, tasks are under active development and undergo testing. They also often wait in queues. We can track their progress.

In a disciplined development organization, every code change is associated with an identified task. Tasks may be requests to implement or change a feature, or defect reports needing resolution. Both kinds of tasks are tracked with task or bug tracking tools, for example Bugzilla or ClearQuest. A task tracker records the submission date of the request or defect report, the date work began, and the date of completion. Other information may also be recorded, such as estimated hours of effort. But for our purposes, we only use the task ID, the task classification (new requirement or repair) and the date of submission. As discussed below, we prefer to use other, more objective sources for the remaining dates.

Code development in a disciplined development organization depends on configuration management. No code change occurs without artifacts being checked in to the configuration management system. Each check-in event identifies an artifact, the date and time, and a developer. When a developer starts work on a new task, he or she checks in artifacts associated with that task. As work progresses, more check-ins occur. When work is completed, no more artifacts are checked in for that task. In order to correlate events, and for us to reconstruct the development history, the event record must include a reference to the associated task.

While some configuration management tools do not require, or directly support task ID’s, task correlation is important to requirements traceability and should be common practice. The ID can be included within the comment, or within the name of a branch. In the latter case, all task work occurs in its own branch. In the future, we expect configuration management tools to better support correlation with tasks.

To produce process metrics, we extract the task tracker and configuration management event logs, clean them to correct misspellings and remove unimportant events and information, and deposit the result into a database. We then construct a table of artifacts and tasks, where each record represents a single artifact’s involvement in a single task. The record’s fields include the artifact and task identifiers, and the dates of first and last change. We find task begin and end dates by grouping all records by task ID, and taking the dates of earliest and latest change.

These dates are combined with the task tracker submission date to get a complete history.

For defect repair tasks, we are also interested in the date when the defect was created. To assign a date, we use the following assumption: the defect was created by a change to an artifact that was later edited to correct the defect. Using this assumption, for each artifact involved in the repair task, we look at all prior changes for all other tasks that were completed prior to the defect's submission date. Taking a conservative approach, we choose the latest date from the set of candidate changes – the date nearest to the defect report. This approach could be refined with additional information, such as line numbers involved. But for our purposes, this simple conservative approach produces a usable result.

Using this repository data, for each repair task, we know how long it took between when work first began and work finally ended, we know how long the task spent in the defect queue between the time it was reported and work first began, and we infer how long it took between the time the defect was caused and when it was found,

Since we know the events we used as defect causes, we can associate defects with prior tasks. For a given feature task, we not only know the time it took to complete the task, we also have the defects that were caused, how long it took to resolve those defects, and even dates of later defects created in the first round of defect repair work.

We measure effort in developer days. A developer is assumed to be active on a task on every day between the first and last change events that associate that developer with that task. We then count each developer as active on any day on which they are active on at least one task. They are counted as actively developing features if they are active on a request task, and they are counted as actively repairing defects if they are active on a repair task. It is possible that the same developer is active on two or more tasks on the same day, and even on one task of each kind on the same day. Our data takes this into account. On the projects for which we have collected data, both situations do occur, but not very often.

#### IV. VISUALIZATION

##### A. Effort.

Figure 1 shows an idealized graph of effort over time. In an ideal world, a team of programmers is assigned to a project and starts working as soon as requirements are available. Work then continues at a steady pace until all of the requirements are completed, with minimal taper at the beginning and the end. The graph would essentially have a boxcar shape. Real projects, of course, do not look like the graph in Figure 1. They look, instead, more like the graph in Figure 2. Developers must be freed from other projects or hired new at the beginning. Some tasks are likely to drag out at the end, keeping smaller numbers

of developers busy beyond the end. Defects are found as the product takes shape and new effort must be devoted to fixing these defects. Testing itself takes time. Some defects require more testing than others before they are found. As depicted in Figure 2, effort ramps up as developers become available and peaks somewhere in the middle. A second later hump of effort addresses rework after tests. The overall effort may not have as pronounced a second hump as depicted in Figure 2, since the later efforts reflect both the rework and requirements that either dragged on or were started late.

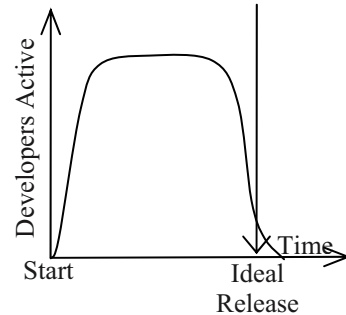


Figure 1. Projects effort vs. time in an ideal process

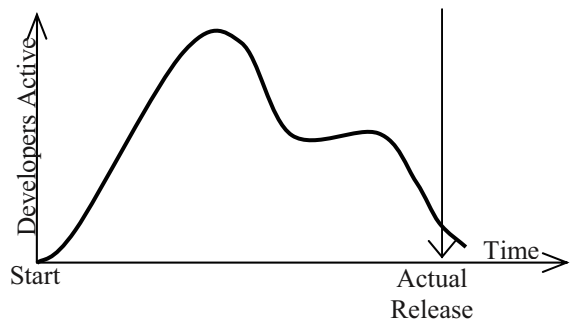


Figure 2. Project effort vs. time with repairs and delays.

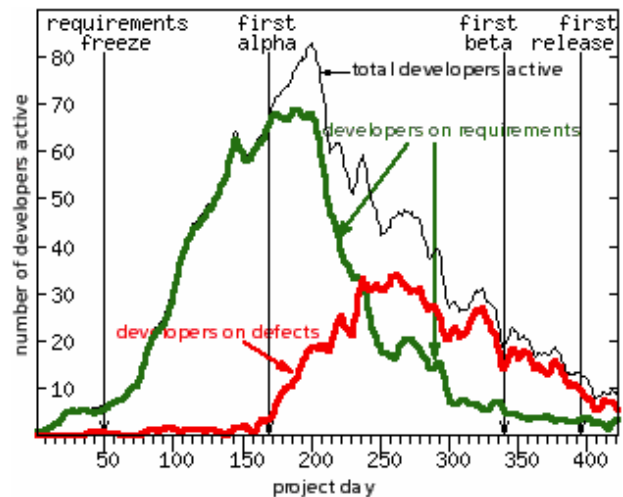


Figure 3. Effort vs. calendar day from real data

Figure 3 shows the real graph of effort vs. time for a project that lasted more than a year. The vertical axis shows the number of developers active on any given day. The horizontal axis shows the day, counted in number of days from the beginning of the project. The project followed a waterfall process; we indicate the dates of the waterfall milestones for requirements freeze, alpha testing release, beta release, and first customer release. The thin line on top shows overall effort, while the two thicker lines show effort separately for requirements and repairs.

Total project effort, measured in developer days, is the area under the curve. We compute it by summing the developers active on each day. The X axis in Figure 3 is actually calendar day. We exclude weekends and holidays from the sum. While total effort differs from one project to the next, the ratio between requirements effort and repair effort is more constant and can be used as a baseline for comparison in process improvement. In this project, 33% of the total effort is devoted to repair. In comparison with other waterfall projects in published literature, and our own experience, 33% is common, and actually better than average.

At a more detailed level, we know that developers do more than code on any given day. But in aggregation, differences in activity pattern of one developer from one day to the next average out and appear the same from one project to the next.

*B. Cumulative Progress*

For measuring progress, we use task as a unit of progress. A project is 50% complete when 50% of the tasks have been completed. We use 5 separate metrics for measuring progress: number of requirements tasks begun, number of requirements tasks completed, number of defects found, number of repair tasks begun, and number of repair tasks completed. From the dates we derived for when each defect was caused, described earlier, we can construct a sixth progress metric: number of defect created.

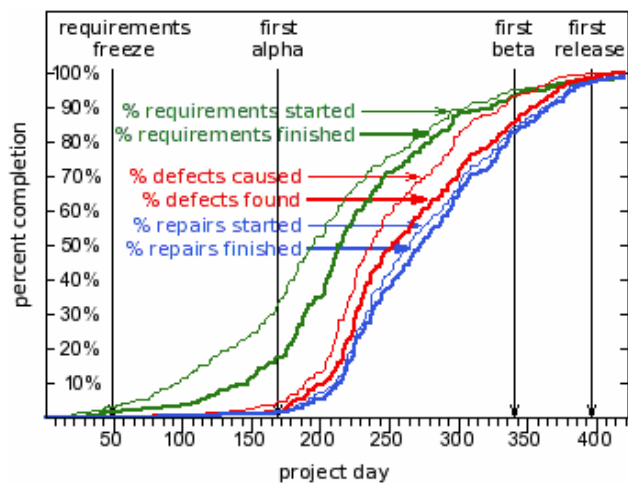


Figure 4. Progress curves from real data

We can plot proportional completion by dividing the number of tasks completed by the total number of tasks in the project. We plotted all 6 curves on the same graph in Figure 4. From left to right the curves are percent of requirements tasks begun, percent of requirements tasks completed, percent of defects created, percent of defects found, percent or repairs begun, and percent of repairs completed. The plots are classic S-curves.

Our conservative estimate of the defect creation events probably puts the “defects caused” curve to the right of where it should be. The left two curves in Figure 4 are measured in terms of requirements tasks, while the remaining four curves on the right are measured in terms of repair tasks. We must be careful not to give meaning to the distance between the two groups – between the 2nd and 3rd curves. They may in fact cross.

The slope of a curve indicates the rate of progress at that point in time. Variations can indicate events or problems. We don’t see a significant uptick in defects being found until day 205 – 4 weeks after alpha testing supposedly began. A significant number of requirement tasks were completed around day 210, which corresponds to a time of developers being released in Figure 3.

*C. Little’s Law*

From queuing theory, Little’s Law tells us that the vertical distance between two curves in the progress graph is a good indicator of the amount of work, or in our case the number of tasks, currently in that phase of the process. This measure is called work-in-process. The vertical distance between the ‘defects found’ curve and the ‘repairs started’ curve is an indicator of the number of defect tasks waiting in the defect queue. The vertical distance between the ‘defects caused’ curve and the ‘defects found’ curve is an indicator of the number of defects in testing waiting to be found.

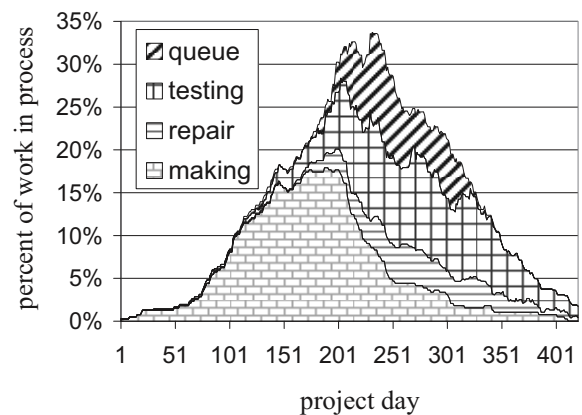


Figure 5. Little’s Law work in process

Figure 5 shows the Little’s Law distances for work-In-process plotted in Excel as an area chart. The area at the bottom is work on requirements, which is distance from the requirements ‘requirements started’ curve to the

'requirements finished' curve. Above that we show work on repairs, computed as the distance from the 'repairs started' curve to the 'repairs finished' curve. Above the repair work we show the defects undergoing test, from the distance between the 'defects caused' and 'defects found' curves. The area at the very top is for defect reports sitting in queue, measured between the 'defects found' curve and 'repairs begun.' The order, bottom to top, was chosen to put the most productive activities on the bottom and the least productive phase at the top.

In Lean and Agile practices, it is considered good to minimize the amount of work in process at any given time. It is hard to make changes when much of the work is midstream in process. A project will be more agile if there is less work in process to better accommodate change.

Little's Law also tells us that the horizontal distance between two curves in the process graph is a good indicator of the amount of time it takes a piece of work to make it through phases of the process. This measure is called time-in-process. The horizontal distance between the defects found curve and the repairs started curve tells us defect reports commonly spend 5 to 10 days in the defect queue. Since we have the actual dates for every repair task, we were able to confirm that that is true.

Figure 6 shows the Excel area chart for Little's Law time in process. It is common for work that takes the longest to be completed at the end.

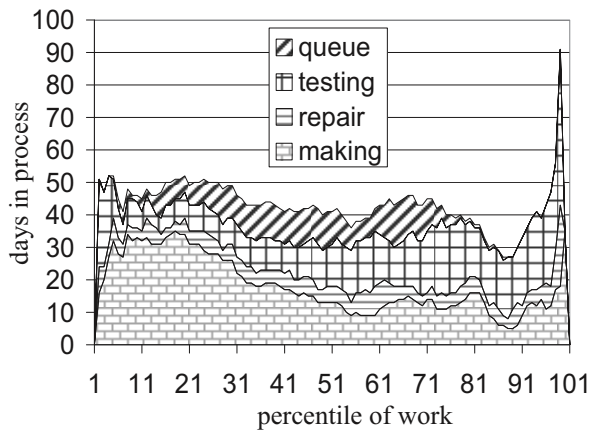


Figure 6. Little's Law time in process.

By Little's Law, the entire distance from the top to the bottom is the time it takes a piece of work to make it all the way through the process, from when work initially started to when the defects are repaired and it is ready to be in the finished product. This measure is called throughput time. The shorter the throughput time, the more responsive the process can be to new or changing demands.

From the repository event data, not only were we able to extract dates for every requirement task and defect repair, but also to link downstream defects to prior tasks.

For a given requirement task, we can assign the repairs it took to get it right. Thus we have real process time for every requirement request. Since there is little uniformity among tasks, the real plot is a lot noisier than that produced by Little's Law. Because the data is from an industry project and contains proprietary information, we cannot publish the real per-task graph. But the average throughput time, computed directly from the per-task data, was 45 days. This value is consistent with the Little's Law time in process graph.

#### D. Progress Path

Vanderwall recently presented a graph of progress that he calls the Project Progress Viewer [18]. The PPV plots completed functionality, on the X axis, against tests passed, on the Y axis. The resulting graph shows the process' path history. The points along the path have time values which also give velocity, and can be used to project future progress.

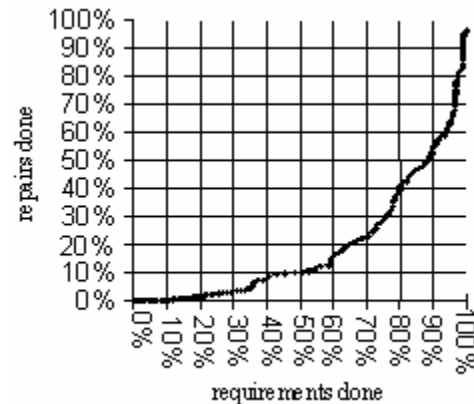


Figure 7. Progress path of requirements vs. repairs.

The shape of the curve in Figure 7 is classic waterfall. Requirements make early progress while defect repair lags significantly behind. In a Lean or Agile process, the graph will present more of a straight line. The project here made good mid-period progress, but slowed considerably towards the end. Not that if the project isn't done, actual numbers can be used in place of percentages, as Vanderwall does.

#### V. CONCLUSION AND FUTURE WORK

In this paper we presented a method of extracting process metrics from common repository data. The data was collected without questionnaires, without adding instrumentation to tools, and without asking developers to provide any information other than what is already provided as part of normal practice. We also showed six basic process analysis graphs that can be drawn from that data. Our intention is to show how objective process metrics can easily be obtained and to draw attention to the opportunity to analyze the process itself. An actual

analysis will depend on the goals of the organization and the types of improvements being considered. We leave that discussion for another paper.

We have done in-depth analyses of several processes including the one shown here. Additional graphs and numerical analyses, beyond those shown here, were used in the analyses. Some details revealed and further investigated in those analyses are visible in the graphs shown here, but are not discussed. We hope to publish that work soon.

The work presented here was started for the purpose of doing Six Sigma analyses for process improvement[19]. We also have related work on the role of artifact and architecture hotspots in process issues, using the same repository data [20].

#### BIBLIOGRAPHY

- [1] S. Huang, S. R. Tilley, M. VanHilst and D. Distanto, Adoption-centric software maintenance process improvement via information integration. *13th IEEE International Workshop on Software Technology and Engineering Practice (STEP 2005)*, 2005, pp. 25-34
- [2] R. Dumke, E. Foltin, R. Koeppe, and A. Winkler, *Softwarequalität durch Meßtools: Assessment, Messung und instrumentierte ISO 9000*. Vieweg Press, 1998.
- [3] B. Boehm and M. Turner. Management challenges to implementing agile processes in traditional development organizations. *IEEE Software*, 22(5) 2005, pp. 30-39
- [4] M. Niazi, D. Wilson, and D. Zowghi. A maturity model for the implementation of software process improvement: an empirical study. *Journal of Systems and Software*, 74(1) January 2005 pp. 155-172.
- [5] A. Rainer and T. Hall, A quantitative and qualitative analysis of factors affecting software processes, *Journal of Systems and Software*, 66(1) April 2003, pp. 7-21.
- [6] S. Zahran. *Software process improvement: practical guidelines for business success*. Addison-Wesley, 1998.
- [7] K. El Emam and L. Briand. Costs and benefits of software process improvement. Technical Report ISERN 97-12, Fraunhofer Institute for Experimental Software Engineering Engineering, 1997
- [8] F. Pettersson, M. Ivarsson, T. Gorschek, and P. Öhman. A practitioner's guide to light weight software process assessment and improvement planning. *The Journal of Systems and Software*. 81(6) June 2008, pp. 972-995.
- [9] Y. Zhang and D. Sheth. Mining software repositories for model-driven development. *IEEE Software*, (23)1:82-90.
- [10] D. Hartman and R. Daymond. Appropriate agile measurement: Using metrics and diagnostics to deliver business value. *IEEE Agile Conference*, 2006, pp. 126-134.
- [11] M. Poppendieck and T. Poppendiech. *Implementing lean software development: From concept to cash*. Addison-Wesley, 2007.
- [12] D.G. Reinertsen. *Managing the design factory: A product developer's toolkit*. The Free Press, 1997.
- [13] P. Johnson, H. Kou, M. Paulding, Q. Zhang, A. Kagawa, and T. Yamashita. Improving software development management through software project telemetry, *IEEE Software*, 22(4) July 2005, pp. 76-85.
- [14] S. Kim and E.J. Whitehead, Jr. How long did it take to fix bugs? *International Workshop on Mining Software Repositories*, 2006, pp. 173-174
- [15] V. R. Basili and H. D. Rombach. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6, June 1988, pp. 758-773.
- [16] R. W. Selby, A. A. Porter, D. C. Schmidt, and J. Berney. Metric-driven analysis and feedback systems for enabling empirically guided software development. *International Conference on Software Engineering*, 1991, pp. 288-298.
- [17] The Perfect Consortium. APEL abstract process engine language. Perfect Handbook Booklet, European Esprit Project, 1997.
- [18] R. Vanderwall. The chart that saved the world. *Software Test and Performance*. (6)1 January 2009, pp. 8-1.
- [19] M. VanHilst, P.K. Garg, and C. Lo. Repository mining and Six Sigma for process improvement. *International Workshop on Mining Software Repositories*, 2005, pp. 1-4.
- [20] S. Huang and C. Lo. Analyzing configuration management repository data for software process improvement, *IEEE International Conference on Software Engineering and Knowledge Engineering (SEKE)* 2007.



# Collaborative Development of System Architecture - a Tool for Coping with Inconsistency

Peter Henderson  
Electronics and Computer Science  
University of Southampton  
SO17 1BJ, UK  
p.henderson@ecs.soton.ac.uk

Matthew J. Henderson  
Mathematics and Computer Science  
Berea College  
Berea, KY 40404, USA  
matthew\_henderson@berea.edu

## Abstract

*Very large systems have an architecture that is designed to allow them to evolve through a long life. Such systems are developed by teams of architects. One of the first things the architects do is make a model of their architecture. This model constitutes the formal architecture description based on which software engineers will eventually build the real system.*

*The architecture model is normally governed by a specialised metamodel whose rules determine the consistency and completeness of the description. The development of a system architecture is carried out cooperatively but independently by team members. Consequently it is quite normal for the architecture description as a whole to be both incomplete and inconsistent. The architects strive to eventually produce a complete overall (i.e. merged) description and to eliminate the inconsistencies.*

*By means of an example, we show how and why the architecture model and the metamodel must co-evolve. We describe a design tool that we have developed to support this process of co-evolution. The tool allows a team of architects to detect inconsistencies in their separate and merged models. The tool tolerates inconsistencies. It produces reports of inconsistencies which then become targets for removal as the whole architecture description evolves.*

## 1. Introduction

Very large systems have an architecture that is designed to allow them to evolve through a long life. The usual way in which such large systems are developed is by first making a model of their architecture. The language chosen for the architecture model is usually a mixture of diagrams (in UML or SysML, for example) and lots of documentation of requirements and of interfaces to the components that are to

be either procured or built. The model is thus a semi-formal architecture description.

The architecture is normally developed incrementally and independently by a team of system architects working collaboratively. While each may strive to keep their part-model of the evolving architecture consistent, there will be inconsistencies between their independent descriptions. These inconsistencies will need to be detected and resolved when, from time to time, the models of independent architects are merged.

The more formal the architecture description, the more likely we are to be able to determine incompletenesses and inconsistencies at an early stage. A formal architecture model is governed by a specialised metamodel whose rules determine the consistency and completeness of the description. During development of a system, it is quite normal for the architecture description to be both incomplete and inconsistent. The architects strive to produce a complete description and to eliminate the inconsistencies.

We describe a method of formalising the rules for the development of a new architecture, in a metamodel that the architects team can agree on, and which can co-evolve with the architecture description itself.

By means of an example, we show how architecture descriptions formalised in this way can aid the iterative process of architecture development and how the model and the specialised metamodel can co-evolve.

We then describe a design-support tool, WAVE, that we have developed to support this process of co-evolution. This tool will calculate inconsistencies within individual and merged models. It does not insist on the architecture model always being consistent. Rather it produces reports of inconsistencies. These inconsistencies are targets for the architects to eventually remove. This means of tolerating inconsistency supports both incremental and collaborative working, essential to the development of large systems by teams of engineers.

## 2. Background

System Architecture is an essential aspect of the design of large system. It forms the overall structure within which the components of the system interact with each other [19, 20, 23, 29] and consequently the basis on which architects negotiate with each other about how the system as a whole will eventually work.

There have been many approaches to the description of System Architecture, both formal and semi-formal. We have been influenced by both, but in particular the more pragmatic methods [1, 16, 17, 18], in particular those that combine familiar semi-formal methods with an element of evaluation [5, 10, 28, 30].

We are particularly concerned with methods that scale up to be applicable to very large systems [2, 12, 14, 20, 22], by which we mean those that will eventually require a large team of software engineers working over an extended period of time. Such methods necessarily involve a great degree of collaboration [8, 21, 25].

Large systems and collaborative development include long periods when the design is both incomplete and inconsistent. The inconsistencies arise when separate parts of the architecture description are developed independently. Many others have worked on the issue of ensuring consistency [6, 9, 26, 27], while others have addressed the issue of tolerating inconsistency [2, 22]. This work has been fundamental in our development of the method and tool that we propose here.

We have also been influenced in the development of our tool, by the tools developed by others, in particular those based on a relational model of architecture [3, 4, 11, 17]. The relational algebra [7, 17, 18] is an ideal formal language for giving structural architecture descriptions and goes a long way towards being appropriate for behavioural descriptions. This, we believe, is because at the level at which architecture description needs to be performed (sufficiently detailed but appropriately abstract) a relational model introduces just the right level of formality. Note that UML (and SysML) have metamodels which are described relationally. This perhaps explains why relational models are a good fit to the task of architecture description.

In the discussion at the end of the paper we mention additional areas of application, including documentation [15] and modular reasoning [24, 13], both of which require architectural support. But first, we illustrate our method using familiar examples from software engineering.

## 3. A Method of Architecture Description

System Architects, building software intensive systems, start from a mixture of user requirements, system requirements and legacy components and devise an architecture

1.	The architects agree a preliminary metamodel, including entities, relationships and consistency rules.
2.	Each architect develops their part of the model, obeying as nearly as possible the current metamodel.
3.	Each architect strives to drive out inconsistencies in their part of the model.
4.	Periodically, models are merged so that cross-model inconsistencies can be eliminated.
5.	Periodically, the metamodel is evolved to encode the architects' evolving understanding of the problem domain.

**Table 1. Architecture Development Method**

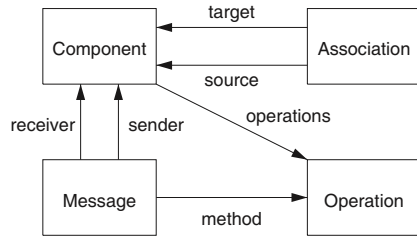
that meets the requirements while making effective use of existing components. They will describe the architecture using a mixture of diagrams and natural language that is effective as a means of communication among them and their customers

Diagrams are most effective at indexing a description. The reader uses the diagrams to get an overview of the (part of the) architecture in which they are interested and then refers to a natural language description to learn the details. The reader will expect to find redundancy in the descriptions and consistency between related parts. For example, in the next section you will see (Figures 2 and 3) diagrams that exhibit redundancy and consistency - in this case, a class diagram and an apparently consistent sequence diagram .

An essential adjunct to the diagrams-plus-natural-language presentation are the consistency rules that the architecture will obey. A judicious use of formal language can complement these necessary aspects of presentation. So the language we choose, to describe a proposed architecture, needs to be sufficiently formal that some consistency checking can be done but not so detailed that the work of describing the architecture is as costly as building the whole system.

Hence many system architects use diagrammatic notations such as those that constitute UML and SysML and specialise them to their specific needs. This specialisation can be represented by a *specialised metamodel* which enumerates the entities that will be used to describe the architecture and defines the constraints that instances of these entities must obey.

The architecture development process that we advocate is shown in Table 1. It comprises an iterative co-evolution of the architecture model and the specialised metamodel. The architects use the “language” defined in the metamodel to capture the architecture description. Since they work as a team, working independently on parts of the description



**Figure 1. A simple metamodel.**

and then merging their efforts, they will introduce inconsistencies that eventually they will strive to remove. In particular, inconsistencies arise when separately developed model-parts are merged. Sometimes, it is not the model that needs to change to eliminate inconsistencies, but the metamodel. The method of Table 1 covers all these aspects. It is of course an iterative process.

This process is supported by our tool, described in a later section, but can be carried out by a disciplined team using their normal development tools, checking the consistency manually by reading each others' contributions when models are merged. Mechanical checking of consistency requires a more formal approach, such as that supported by our tool. Before introducing that, we will develop a simple example showing an architecture model being developed collaboratively and its metamodel being evolved.

#### 4. An Architecture Example

We consider a team of architects developing a large software system. The system is to be built from components that send messages to each other (probably by a mechanism such as RPC). The first thing the architectural team must determine is their metamodel. Let us assume that they are going to construct a model comprising Class diagrams (or Component diagrams) with associations between the classes recording client/server (i.e. uses) relationships. Let us also assume that they are going to record scenarios (e.g. details of Use Cases) in Sequence diagrams where messages are exchanged between components. The kind of consistencies they might wish to maintain are that messages may only pass from clients to servers and that every operation of a Component will be exercised by at least one scenario.

Examples of the Class diagrams and Sequence diagrams that might be developed are shown in Figures 2 and 3 respectively. In practice we expect such diagrams to contain many more entities than this trivial example and to be split across many separate diagrams. This is why consistency becomes an issue. The full-scale examples that we have used to evaluate our tool have contained on the order of

consistency rule	description
no invalid messages	For each Message from one Component to another there is a corresponding Association in some class diagram.
no untested messages	For each Association from one Component to another there is a corresponding Message in at least one scenario.
no undefined methods	For each Operation appearing on a Message there is a corresponding definition in the receiving Component.
no untested methods	For each Operation defined on each Component there is at least one Message in some scenario that uses it.
no cycles	For the purposes of loose-coupling, there should be no cycles in the Associations established across all Class diagrams.

**Table 2. Some Consistency Rules.**

fifty classes and comprised a few dozen independently developed diagrams.

We will use a relational model to develop the formal aspects of our architecture description as, we have discussed, many others have done before us [4, 11, 17, 18, 27].

The metamodel that the architecture team constructs at the outset might look like that shown in Figure 1 and have the consistency rules enumerated as in Table 2.

The metamodel says that the entities appearing on the diagrams will be Components, Associations, Messages and Operations. The Components and Associations will appear respectively as boxes and arrows on Class diagrams, such as in Figures 2 and 4. The Messages and Operations will appear on Sequence diagrams, such as in Figures 3 and 5. The diagrams record relational information about these entities. For example, the Class diagrams record the fact that each Association has a source and a target, both of which are Components. Similarly, the Sequence diagrams record the fact that each Message has a sender and a receiver (both Components) as well as a method, which is an operation of the receiver. The metamodel which captures these relationships, also assumes that we have an enumeration of the Operations of each Component.

Table 2 shows an initial set of consistency rules that we assume the architecture team have enumerated. These rules are simple, but typical of the structural consistencies that the team will be trying to achieve. Basically, the rules state

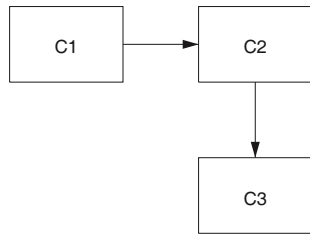


Figure 2. Architect 1's Class Diagram.

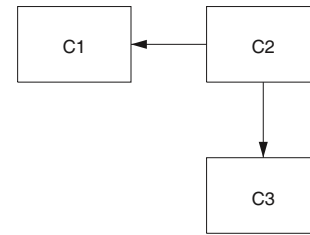


Figure 4. Architect 2's Class Diagram.

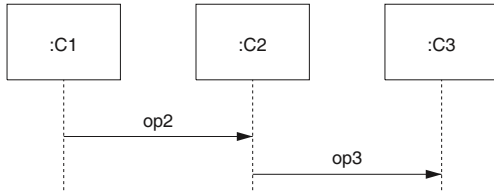


Figure 3. Architect 1's Sequence Diagram.

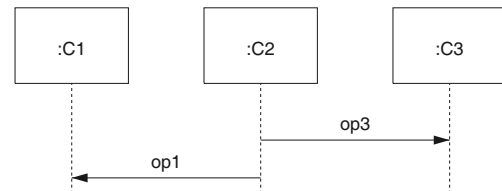


Figure 5. Architect 2's Sequence Diagram.

that every Message should be from client to server, that every Message should be tested in some scenario and that (for loose-coupling) cycles in the Class diagrams are to be avoided. We will see that, while these rules can be obeyed, the architectural team eventually chooses to relax them (and consequently evolve the metamodel).

## 5. Consistency

Having agreed the metamodel and its rules, the team then begins independent development of separate parts of the architecture. Suppose Architect 1 comes up with the Class diagram in Figure 2 and the Sequence diagram in Figure 3. These two diagrams almost satisfy the five rules of Table 2. In fact, our tool notes that there is an operation of C1 which is not tested (op1), but that is the only problem. Architect 1 is satisfied with this, because it is someone else's task to exercise C1.

In practice, when an individual architect is working on a part of an architecture, they may be dealing with a few dozen Components and developing (say) a dozen or more scenarios. Arriving at an acceptable level of internal consistency will be an iterative process. Residual inconsistencies will be (the architect hopes) resolved when their part is combined with the other parts being developed concurrently.

Architect 2 has, we imagine, concurrently developed the Class diagram shown in Figure 4 and the Sequence diagram shown in Figure 5. They receive much the same report as Architect 1 - all is consistent, except there is an operation of C2 that has not been tested (op2).

However, when the two architects combine their models and run consistency checking again, they encounter a little difficulty.

While the inconsistencies that they knew about when developing independently are resolved by the combination of models (now, all known Operations are tested), the combination of the Class diagrams has unfortunately created a loop in the Associations between C1 and C2. This is something that will need to be resolved, either by one of the architects making modifications to their contribution, or by a change to the metamodel.

The reason we are so concerned with inconsistency is that, for large systems, many components and many scenarios will be defined and it will considerably improve the quality of the description if these individual descriptions are eventually made consistent, while for pragmatic reasons we must tolerate inconsistency during development.

In our example we have a choice of the remedial action to take. It may be possible for one architect to change their model. If not, the team will consider if the metamodel needs to be evolved.

In fact, here, we have probably made the loose-coupling constraint too strong. We might allow "local" loops within well defined subgraphs of the whole architecture. Here for example, it would be sufficient to allow loops between Components which are of length no greater than two. If A uses B, then allowing B to use A does not really damage the coupling, we could argue (a common feature in implementations that use some form of callback).

In practice, we will wish to construct more complex relations than those that we have exemplified here and for which

we will require more expressive forms than the simple relational diagrams used in this section. In the next section we describe how we have based our tooling on the relational algebra, and how this both simplifies the encoding of the consistency rules and the evolution of the metamodel.

## 6. A Design Support Tool

The WAVE tool has been implemented to support the way of working outlined in Table 1 and illustrated in the previous section. The architects work concurrently on separate (overlapping) parts of the architecture, recording their progress in a local copy of a “database”. In practice, we generate this database from whichever diagramming tool the architects choose to use, by dumping it as an XMI file and importing it into the WAVE tool.

In the implementation we shall describe here (available from <http://ecs.soton.ac.uk/~ph/WAVE>) the database is held as a set of relational tables and the consistency rules are implemented as scripts that compute new tables recording discovered inconsistencies.

The use of a relational model and of scripts to define the consistency rules makes the co-evolution of the metamodel particularly adaptable and straightforward.

For example, the script that computes the first consistency rule in Table 1 is written

```
invalid_messages =
  diff(join(invert(sender), receiver),
        join(invert(source), target))
```

Here, `join` is the relational join of two (binary) relations. The relations are those recorded in the metamodel (Figure 1) and the data in them records that displayed on the actual model diagrams. The operation `invert` takes the inverse of a (binary) relation and `diff` computes the set difference. So the above calculation constructs a relation which relates any two Components between which there is a Message (on some sequence diagram) but between which there is no corresponding Association on any Class diagram. This computed relation is effectively an inconsistency report - listing those places where rule 1 is disobeyed.

The fact that WAVE is scripted and stores its data in relational tables (exportable as tables or as XML) means that generating reports of inconsistencies is also straightforward. Architects can thus work with independent copies of the database and work to extend their view of the architecture and to remove inconsistencies that are reported.

When databases are merged in WAVE the simplest, and apparently most effective, merge is simply to take the union of each copy of each relational table. WAVE has been specifically designed so that for most purposes this is the most effective merge. It does mean however that, if an entity appears in two copies of the database and is deleted by only

one of the architects, it will reappear on merge. If it was required that deletes by one architect would propagate on merge, then a more elaborate (diff3-like) merge is required. Since all operations on the database are scripted, including merge, making a domain-specific merge is as straightforward as any metamodel evolution.

We have used WAVE on a number of small projects and a couple of fairly large ones. The largest has about fifty classes (actually, components) spread over about a dozen class diagrams and another dozen sequence diagrams. In the current version of the architecture described in this largest example, WAVE lists about twenty inconsistencies that are genuine inconsistencies between model-parts developed by different architects and a smaller number that are probably going to require the metamodel (which includes the metamodel shown here as a subset) to be co-evolved. Running the WAVE scripts against the XMI files for this architecture (dumped, as it happens, from Sparx Systems’ Enterprise Architect) takes a few seconds. We have tested performance on an artificial example with around 1000 entries in each of the relational tables that are generated as intermediate structures in WAVE, to persuade ourselves that WAVE’s performance will scale to realistic large-scale models. These tests have shown that WAVE can process such tables with constraints such as those listed in Table 2 in seconds rather than minutes. A more comprehensive performance analysis will be completed soon.

## 7. Conclusions

We have described a method of developing architecture descriptions based on giving a sufficiently precise metamodel that consistency can be checked during architecture development.

Rather than insist that the architecture description is kept consistent at all times, we advocate a method of iterative and cooperative development that allows the description to be periodically inconsistent.

We have shown how the metamodel can be captured formally as a relational model and argued that this method is particularly appropriate to this style of development, not least of all because it encourages the architects to embrace the whole architecture at all times and to keep in mind how far from internal consistency the description may have drifted.

The tool we have described supports this method of working, where model and metamodel are co-evolved and where inconsistency is tolerated during development. For large scale systems, where iterative and cooperative working is the norm, this tolerance of inconsistency is essential.

The method has been applied so far only to software intensive systems. It also seems appropriate to other domains. We have in hand experiments with metamodels for docu-

mentation [15], for reasoning [24, 13] and have an ambition to extend the method to a broader range of systems, in particular those that have physical as well as logical structure. Ultimately our plan is to combine these domains so that describing an architecture, documenting it and reasoning about it will all be supported within the same framework.

## References

- [1] K. Balasubramanian, J. Balasubramanian, J. Parsons, A. Gokhale, and D. C. Schmidt. A platform independent component modeling language for distributed real-time and embedded systems. *J. Comput. Syst. Sci.*, 73(2):171–185, 2007.
- [2] B. Balzer. Tolerating inconsistency. In *ICSE '91: Proceedings of the 13th international conference on Software engineering*, New York, NY, USA, 1991. ACM.
- [3] D. Beyer. Relational programming with CrocoPat. In *ICSE*. IEEE, 2006.
- [4] D. Beyer, A. Noack, and C. Lewerenz. Efficient relational calculation for software analysis. *Transactions on Software Engineering*, 31(2):137–149, 2005.
- [5] K.-N. Chang. Consistency checks on UML diagrams. In *International Conference on Software Engineering Research and Practice, SERP07*. IEEE, 2007.
- [6] K.-N. Chang. Model checking consistency between sequence and state diagrams. In *International Conference on Software Engineering Research and Practice, SERP08*. IEEE, 2008.
- [7] C. Date. *Database in Depth - Relational Theory for Practitioners*. O'Reilly Media Inc., Sebastopol, CA, 2006.
- [8] U. Dekel and J. D. Herbsleb. Notation and representation in collaborative object-oriented design: an observational study. In *SIGPLAN Notices, Volume 42, Issue 10*. ACM, 2007.
- [9] A. Egyed. Instant consistency checking for the UML. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 381–390, New York, NY, USA, 2006. ACM.
- [10] A. Egyed. Fixing inconsistencies in UML design models. In *ICSE '07: Proceedings of the 29th international conference on Software engineering*, New York, NY, USA, 2007. ACM.
- [11] A. Egyed. UML/analyzer: A tool for the instant consistency checking of UML models. In *ICSE '07: Proceedings of the 29th international conference on Software engineering*, New York, NY, USA, 2007. ACM.
- [12] S. Fickas. Clinical requirements engineering. In *In ICSE 05: Proceedings of the 27th International Conference on Software Engineering*, pages 28–34. ACM Press, 2005.
- [13] C. B. Haley, J. D. Moffett, R. Laney, and B. Nuseibeh. Arguing security: Validating security requirements using structured argumentation. In *in Proceedings of the Third Symposium on Requirements Engineering for Information Security (SREIS'05), co-located with the 13th International Requirements Engineering Conference (RE'05)*, 2005.
- [14] P. Henderson. Laws for dynamic systems. In *International Conference on Software Re-Use (ICSR 98)*. IEEE, 1998.
- [15] P. Henderson and N. de Silva. System architecture induces document architecture. In *20th International Conference on Software Engineering and Knowledge Engineering (SEKE 2008)*. IEEE, 2008.
- [16] P. Henderson and J. Yang. Reusable web services. In *8th International Conference on Software Reuse (ICSR 2004)*. IEEE, 2004.
- [17] R. C. Holt. Binary relational algebra applied to software architecture. In *CSRI Technical Report 345*. University of Toronto, 1996.
- [18] D. Jackson. *Software Abstraction*. MIT Press, Cambridge, MA, 2006.
- [19] P. Kruchten. Architectural Blueprints - the 4+1 view model of software architecture. *IEEE Software*, 12(6):42–50, 1995.
- [20] M. W. Maier and E. Reichtin. *The Art of System Architecting, 2nd Ed.* CRC Press LLC, Boca Raton, FL, 2002.
- [21] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of statecharts specifications. In *In 29th International Conference on Software Engineering (ICSE07)*, pages 54–64, 2007.
- [22] B. Nuseibeh, S. Easterbrook, and A. Russo. Making inconsistency respectable in software development. *Journal of Systems and Software*, 58:171–180, 2001.
- [23] N. Rozanski and E. Woods. *Software Systems Architecture*. Addison-Wesley, Upper Saddle River, NJ, 2005.
- [24] J. Rushby. *Modular Certification*. SRI International, Menlo Park, CA, 2002.
- [25] M. Sabetzadeh and S. Easterbrook. View merging in the presence of incompleteness and inconsistency. *Requir. Eng.*, 11(3):174–193, 2006.
- [26] M. Sabetzadeh, S. Nejati, S. Easterbrook, and M. Chechik. Global consistency checking of distributed models with tremor. In *In 30th International Conference on Software Engineering (ICSE08), 2008. Formal Research Demonstration (To Appear)*.
- [27] M. Sabetzadeh, S. Nejati, S. Liaskos, S. Easterbrook, and M. Chechik. Consistency checking of conceptual models via model merging. In *In RE*, pages 221–230, 2007.
- [28] M. Shaw and P. Clements. The golden age of software architecture. *Software*, March/April:31–39, 2006.
- [29] M. Shaw and D. Garlan. *Software Architecture - Perspectives on an emerging discipline*. Addison-Wesley, Upper Saddle River, NJ, 1996.
- [30] W. Shen, K. Compton, and J. Huggins. A toolset for supporting UML static and dynamic model checking. In *In Proc. 16th IEEE International Conference on Automated Software Engineering (ASE)*, pages 147–152. IEEE Computer Society, 2001.

# **BITS: Issue Tracking and Project Management Tool in Healthcare Software Development**

Ayşe Tosun, Ayşe Bener, Ekrem Kocaguneli

*Software Research Laboratory, Computer Engineering Department, Boğaziçi University, Turkey  
ayse.tosun@boun.edu.tr, bener@boun.edu.tr, ekrem.kocaguneli@boun.edu.tr*

## **Abstract**

*Healthcare information management systems (HIMS) are critical in day-to-day management of large healthcare institutions to provide timely and accurate patient/diagnosis/treatment information, to improve the quality of service and to lower the costs. Poor implementation of such systems may cause critical failures, such as inaccurate patient records, wrong treatments. It is necessary to prioritize software quality and process management activities during implementation of HIMS. We have worked with a medium size enterprise, which has a HIMS product, to build an in-house Issue Tracking and Project Management Tool. Using this tool, we have managed to a) collect customer requests automatically, b) plan the projects, c) implement software processes, and d) manage the projects in terms of bug tracking, version control and reporting. We have observed that software development effort per a given task has decreased by 82%. Improvements in the quality of service in HIMS have led to increase in customer satisfaction.*

## **1. Introduction**

According to U.S. Labor Statistics, healthcare industry is massive and the fastest growing for the last 10 years, providing 13.5 million jobs to professionals in healthcare [6]. Similarly, a recent report published by Turkish Statistical Institute indicate that healthcare sector has been growing by 11.8% in Turkey, which is the largest growth among other industries in 2008 [5].

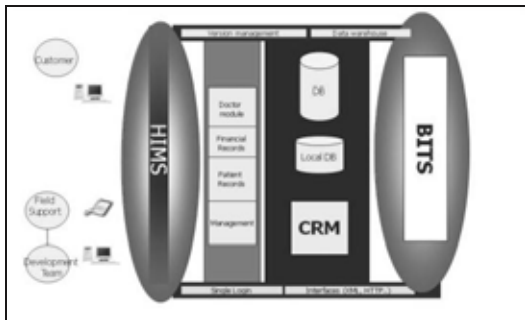
Information management in healthcare systems is vital to improve quality of service, to ensure about the accuracy of critical information and to maximize the efficiency across various units of institutions, while lowering the healthcare expenses [2, 4]. In software development, the ultimate goal is to develop well-defined information management systems that provide efficient workflows, easier processes and reduced errors [1, 2]. However, such systems in healthcare

should be designed even more carefully in order to prevent incorrect information in patients' diagnosis, treatments and major problems in financial records [2]. Implementation of these systems in healthcare needs rigorous effort to control software processes and assure high quality in the software.

Bilmed is a medium size enterprise in healthcare industry in Turkey for 20 years. The company has a fully integrated Hospital Information Management System (HIMS), which manages patient records, their diagnosis, treatments, lab records, medical stocks, financial services and personnel records in hospitals and medical institutions. Their product has been frequently updated due to the changes in government regulations. They need to respond to those changes as quickly as possible to meet deadlines. They need to preserve product quality in terms of reduced defect rates, improved efficiency and customer satisfaction.

A report on performance evaluation of healthcare institutes and process automation in Turkey includes records from 55 hospitals and 14 field studies from public and private healthcare institutes [3]. Among them, only two hospitals, which are using the HIMS, were able to pass the performance evaluation on stock, medical device and information management, and process automation. In 'Hospital Automation' part, these two hospitals are presented as pioneers to realize the benefits of process automation in terms of prevention in loss of income, decrease in total expenses, quick return on investment, improved quality of service, i.e. average waiting time, and access to information.

In 2008, Bilmed decided to measure, manage and evaluate their software development. The company was aware of the fact that they need an issue tracking and project management system to improve the efficiency in development and healthcare processes. We have decided to focus on software processes via an automated issue tracking and project management tool. We have defined the company's needs from this tool as follows:



**Figure 1. Overall architecture of BITS**

- To decrease the documentation effort
- To collect customer requests easily and immediately
- Requirements gathering and analysis
- To monitor the status of a new project
- Project planning, analysis and management
- To control the latest versions at different healthcare institutions
- Traceability of new requests/ failures

Although there are modeling and design tools to complete requirements analysis activities and to trace those with the customer requests [8], senior managers of Bilmed needed a comprehensive issue tracking and project management system that could also be fully integrated to their ERP product. They envision that a system would work as an integral part of HIMS so that nothing changes at the customer side. The aim was to capture faults and new requirements automatically as the doctor/ nurse would be on-line on HIMS. Such a system would also solve on-site problems and perform version control without assigning field support personnel to every customer unit. Therefore, we have initiated the implementation of an in-house tool, namely Bilmed Issue Tracking and Project Management System (BITS). In this paper, we will explain the motivation behind the development of BITS, present fundamental features of BITS and discuss the benefits in terms of software development effort spent per task.

## 2. BITS: Issue Tracking and Project Management System

BITS consists of four different modules: collecting customer requests, project planning, application of software processes and project management. It has been developed in 15 weeks, i.e. (7, 3, 5) weeks for requirements analysis, coding and testing, respectively [7]. It is a web based application that lets the user log into the system through a website (Figure 1). Although software team use a web based application for process



**Figure 2. List of customer requests opened in BITS**  
(Details are shaded for security reasons in all figures.)

automation in BITS, customers use the existing user interfaces from HIMS. BITS works behind the modules of HIMS to trace unexpected failures automatically and to get requests and feedbacks from customers. It operates seamlessly under the generic structure that customers get used to in their healthcare institutions.

### 2.1. Customer Requests

Previously, customers of Bilmed used various ways of communication including e-mail, telephone or separate to-do lists to request their needs. The company tried to identify and classify these requirements via several phone calls or meetings both internally and with the customer. However, users at the hospitals (doctors, nurses, lab managers, support personnel, etc.) have a busy daily routine that requirements gathering and clarification becomes a challenge.

Clarification of ambiguities and misunderstandings cause more effort, time and defects to the customer and software team. Automated tools help customers to express their requests exactly and help us to decrease the complexity of the problem. We have designed BITS such that customers share screenshots or other documents through the system. We have also used BITS to get new requests from the customer who actively uses their HIMS product. When a new functionality is needed or a problem is occurred in the HIMS, the customer, i.e. doctors, nurses or field support personnel, immediately opens a new job on the BITS using the same interface from HIMS (Figure 2). They roughly categorize their request in terms of maintenance, development or bug fixing to help managers for estimating the scope of the work

### 2.2. Project Planning

Managers in Bilmed intend to control time, schedule and man-hour efforts for a project using BITS. They often assign a new job to the developers and then it is shared by development team based on their workload



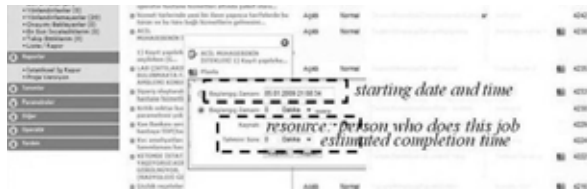


Figure 3. Sample screenshot for project planning

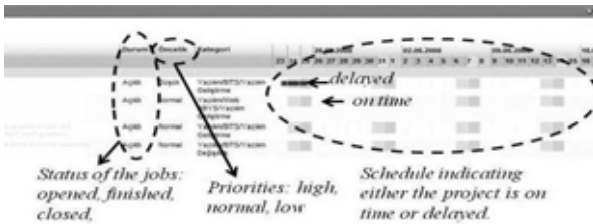


Figure 4. Schedule of a person from software team

and job complexity. Although they somehow plan time/effort for that job, it is hard to control and measure estimation accuracy and personnel performance. Using BITS, it has become easier to plan required tasks, observe if sub-tasks are shared by developers, observe their progress and evaluate estimations.

When a new job is opened by a customer or software manager, it is categorized as either “software development” or “service support”. If possible, software development tasks are classified as “new requirement”, “failure/bug”, “customer misuse” and “deployment request”. If the job is a service support, it is further classified as “database support” or “version support”. An opened job can be planned by adding the estimated time (in hours), required number of personnel from software team (Figure 3). The system provides the user to see the current workload of each personnel, the scope of his/her past projects and the results of these projects for a specific time period, in terms of either finished on time or delayed (Figure 4). Project manager can decide which person from the team has the capability of doing that job by looking at those statistics. After planning, the situation can be controlled day by day from BITS.

### 2.3. Implementing Software Processes

Previously, software team in Bilmed manually followed requirements gathering, design and implementation stages. However, these are not well-documented and deliverables were completed by heroic efforts of few people. Any process during software development could not be monitored and evaluated.

Process automation was inevitable for the company to decrease the documentation effort, make project and process information available to all users and keep all

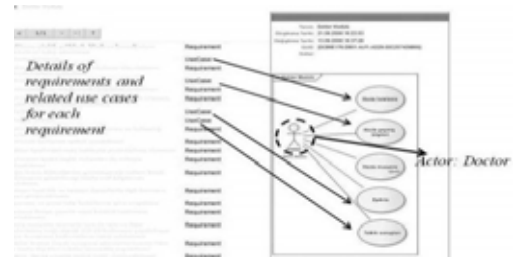


Figure 5. Requirements management activities

activities up to date. We have used a commercial UML design and modeling tool, Enterprise Architect [8], inside BITS. Enterprise Architect (EA) is not only a modeling and design tool, but it also provides complete traceability from requirements analysis to design, implementation and deployment [8]. We have used this tool for adding customer requests as requirements, designing UML diagrams, generating use cases, scenarios, and data diagrams. Then, we have integrated the outputs of EA to BITS. Figure 5 represents detailed requirements and use case diagrams for a specific module of HIMS product. When the job is an enhancement or a change request of an existing product, its completed use cases, diagrams can be accessed over BITS. Using previous information, developers can see the effects of the change on other parts of the system and implement accordingly. They update diagrams to match these with the latest version of product. As a result, we have managed to complete major steps of requirements management. This improved the quality of HIMS, since requirements and possible problems are precisely defined and analyzed before deployment. So, project tracking would be more efficient for the managers.

### 2.4. Project Management

Project management module consists of three parts: automated bug tracing, version control and reporting.

**Automated Bug Tracing:** Some bugs in the HIMS product may not cause interruption or availability of the system. Most of the time, customers ignore such small errors. We have designed BITS to provide for an automated bug tracing mechanism for the bugs that were not traced by the company so far. Whenever an exception in HIMS is occurred, it is automatically written to BITS log. Then, an e-mail is sent to development team with the exact error message including module name and current version of the system. Each personnel in development team can examine BITS logs and resolve that specific problem. Developers in the company state that the automation for finding software bugs or failures in the system has

been very effective to decrease defect rates, to improve software quality and customer satisfaction. They fix problems in a shorter time and avoid further risks in a seamless manner to their customers.

**Version Control:** Version control property is used to show the functionality contents of versions, to control different modules of HIMS that are actively used by various hospitals, and to control database versions that vary across different versions of HIMS. Using this property, we can search all changes which are already deployed or the ones at the implementation stage for a specific version.

**Reporting:** Measurement and analysis is essential for effective management of product quality, planning objectives, estimating costs and taking corrective actions [9]. We have not fully implemented the practices of this process area. However, we have automatically collected essential information, i.e., failures in the system, and stored project attributes such as planned vs. actual time/ effort for a given job and people who actively involved from beginning to end. To analyze this data, we plot the number of “software development” tasks between September and December 2008 (period when BITS is actively used by all customers) and the efforts (man-hours) spent per each task during that period. Results (Figure 6) show that the number of completed software development projects increases except the last month. This shows that customers collaborated with the software team via BITS and they actively used the tool for their requests. On the contrary, the effort spent for each task reduces linearly. In other words, developers spend fewer hours to complete a job and complete more projects monthly.

### 3. Conclusion

We have shared our experience on software process improvement practices in a medium size enterprise that operates in healthcare industry. We have implemented an issue tracking and project management system (BITS) for their company. Using BITS, managers have been able to plan processes, control them and measure some characteristics of projects. They have managed to gather large amount of data related to software processes inside the company and customer requests and feedbacks from the hospitals. Activities such as bug fixing and deployment of latest versions are done invisible to customers. Although there are more issues to be dealt with, BITS presents a successful application of process automation in fundamental software engineering practices in healthcare. Both the software team and customers in healthcare institutions have seen tangible benefits in terms of effort spent for software

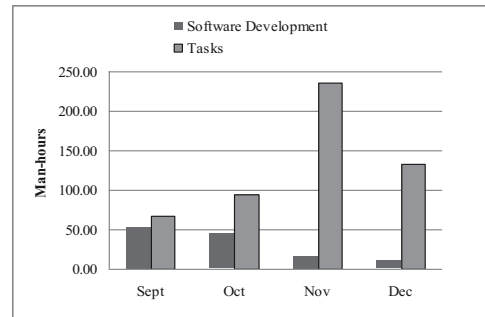


Figure 6. Software development tasks vs. effort

development, deployment and maintenance. These benefits lead efficient and fault-free service in information management systems. Going forward, we would like to further extend BITS to include an intelligent oracle to predict defects and meet resource allocation needs.

### 4. Acknowledgment

This research is supported in part by Turkish Scientific Research Council, TUBITAK, under grant number EEEAG108E014, and Bilmed A.Ş.

### 5. References

- [1] Francisco, J.R., Health Care Software Engineering: A Review of Selected Literature. *Software Engineering in Health Care*, Working Paper, University of Phoenix, March 2004, available at SSRN: <http://ssrn.com/abstract=607446>.
- [2] Glickman, S.W., Baggett, K.A., Krubert, C.G., Peterson, E.D., Schulman, K.A., Promoting Quality: The Health Care Organization From a Management Perspective. *International Journal for Quality in Health Care*, 19(6): 341-348, 2007.
- [3] T.R Court of Accounts, Performance Audit Report, March 2005.
- [4] Haux, R., Winter, A., Ammenwerth, E., Brigl, B., Strategic Information Management in Hospitals: An Introduction to Hospital Information Systems. *Springer*, 2004.
- [5] Turkish Statistical Institute, Industrial Growth Report for the First Quarter of 2008, 2008.
- [6] Bureau of Labor Statistics, U.S. Department of Labor, Career Guide to Industries. 2006-07 Edition, [www.bls.gov](http://www.bls.gov).
- [7] Tosun, A., Turhan, B., Bener, A., The Benefits of a Software Quality Improvement Project in a Medical Software Company: A Before and After Comparison, *Invited Paper in International Symposium on Health Informatics and Bioinformatics*, 2008.
- [8] Enterprise Architect UML Modeling Tool, available at [www.sparxsystems.com.au](http://www.sparxsystems.com.au), 2008.
- [9] CMMI for Development, Version 1.2, Carnegie Mellon Software Engineering Institute, August 2006.

# Privacy-Preserving Clustering of Data Streams

Ching-Ming Chao and Chih-Chin Shen

Department of Computer Science and Information Management

Soochow University, Taipei, Taiwan

## Abstract

Recently, data streams are emerging as a new data type. Traditional privacy-preserving data mining techniques are not suitable for data streams. Most studies on privacy-preserving data stream mining focus on association analysis and classification. In this paper, we propose a method called PPCDS for privacy-preserving clustering of data streams. PPCDS is composed of two phases: rotation-based perturbation and cluster mining. In the first phase, a rotation transformation matrix is applied to rapidly perturb data streams in order to preserve data privacy. In the second phase, perturbed data first establish micro-clusters through optimization of cluster centers, then apply statistical calculation to update micro-clusters, as well as using geometric time frame to allocate and store micro-clusters, and finally obtain mining results through macro-cluster generation. This process reduces repeated calculation time to enhance mining efficiency without losing mining accuracy.

## 1. Introduction

Nowadays it is important to find out useful information from massive amounts of data. Consequently, various data mining techniques have been developed. Besides, the rapid advance in Internet and communications technology has led to the emergence of data streams. Therefore, the study of data mining techniques has transformed from traditional static data mining to dynamic data stream mining.

In recent years, many companies enhance competition through strategic alliance or information sharing. They expose private data while engaging in data analysis activities, which leads to great threat to data privacy. Therefore, how to preserve data privacy and also obtain accurate mining results becomes a challenge, which leads to the development of privacy-preserving data mining techniques [1]. Traditional algorithms for privacy-preserving data mining are not suitable for the data stream environment. Therefore, privacy-preserving data stream mining has become one of the important issues in the field of data mining.

However, most of the studies on privacy-preserving data stream mining focus on association analysis and classification. Only a few studies focus on clustering [2,3,4]. In this paper, therefore, we propose a method called PPCDS for privacy-preserving clustering of data streams. PPCDS is composed of two phases: rotation-based perturbation and cluster mining. In the first phase, a rotation transformation matrix is employed to rapidly perturb data streams in order to preserve data privacy. In the second phase, perturbed data first establish micro-clusters through optimization of cluster centers, then apply statistical calculation to update micro-clusters, as well as using geometric time frame to

allocate and store micro-clusters, and finally obtain mining results through macro-cluster generation. Two simple data structures are added in the macro-cluster generation process to avoid recalculating the distance between the macro-point and the cluster center. This process reduces the repeated calculation time in order to enhance mining efficiency without losing mining accuracy.

## 2. The PPCDS Method

The PPCDS method is mainly composed of two phases: rotation-based perturbation and cluster mining.

### 2.1 Rotation-Based Perturbation

When a data stream is incoming, data is represented in an  $m \times n$  data matrix  $D_{mn}$ , in which each row represents one entry and each column represents an attribute of data. Subsequently a rotation transformation matrix  $R(\theta)$  is allocated to perturb data, with the data on the coordinate axis is rotated clockwise in a  $\theta$  angle in order to perturb data. Figure 1 shows the process of rotation-based perturbation with parameters defined below:

- $D_{mn}$  represents the pre-perturbed data, while  $D'_{mn}$  represents the post-perturbed data.
- $T$  represents the number of unperturbed attributes.
- $\theta$  represents the perturbing angle set up by the user.
- $R(\theta)$  represents the rotation transformation matrix for perturbing angle set up by the user.
- $A_j$  and  $A_k$  ( $1 \leq j, k \leq n, j \neq k$ ) represent the attributes selected from  $D_{mn}$  for perturbation.
- $V(A_j, A_k)$  represents the data with pre-perturbed attribute, while  $V'(A_j, A_k)$  represents the data with post-perturbed attribute.

Detailed steps are described below:

Step 1: Set the initial value of unperturbed attribute number  $T$  as the attribute number  $n$  of data matrix  $D_{mn}$ .

Step 2: Determine the existence of any unperturbed attribute, if affirmative then execute the loop.

Step 2.1: In the event of more than one unperturbed attribute, randomly select two attributes,  $A_j$  and  $A_k$ , from  $D_{mn}$  to perform rotation perturbation on selected attributed data  $V(A_j, A_k)$ , using Rotation Transformation Matrix  $R(\theta)$  and to reduce  $T$  value by 2.

Step 2.2: In the event of only one unperturbed attribute,

randomly select an already perturbed attribute  $A_j$  and the remaining last attribute  $A_k$  to perform perturbation, and reduce T value by 1.

<b>Input :</b> $D_{mn}, \theta$
<b>Output :</b> $D'_{mn}$
<pre> 1. T ← n; 2. While (T &gt; 0) do { 2.1 If (T &gt; 1)     { Randomly select <math>A_j</math> and <math>A_k</math> from <math>D_{mn}</math>;       <math>V'(A_j, A_k) \leftarrow R(\theta) \times V(A_j, A_k)</math>;       T ← T-2;}   2.2 Else // T = 1     { Randomly select an already distorted attribute <math>A_j</math> with the last attribute <math>A_k</math> from <math>D_{mn}</math>;       <math>V'(A_j, A_k) \leftarrow R(\theta) \times V(A_j, A_k)</math>;       T ← T-1;}}</pre>

**Figure 1. Rotation-based perturbation process.**

## 2.2 Cluster Mining

### 2.2.1 Micro-Cluster Generation

A micro cluster is an extension of cluster feature vector, with a main purpose of recording statistical information of data points after rotation perturbation. Assume one micro-cluster represents  $n$  number of multidimensional data  $\overline{X}_1 \dots \overline{X}_n$ , each multidimensional data is represented by  $\overline{X}_i = (x_i^1 \dots x_i^d)$ , and each multidimensional data has its proprietary timestamp  $T_1 \dots T_n$ . Each micro-cluster has  $(2 \times d + 3)$  numbers of data items, with  $d$  as the attribute number and expressed as  $\{\overline{SS}, \overline{TS}, SST, ST, n\}$ . Among which,  $\overline{SS} = \{SS_1, SS_2, \dots, SS_p, \dots, SS_d\}$ ,  $SS_p = \sum_{i=1}^n (x_i^p)^2$ ,  $1 \leq p \leq d$ , which is the total square sum for data value of  $p$ -th attribute;  $\overline{TS} = \{TS_1, TS_2, \dots, TS_p, \dots, TS_d\}$ ,  $TS_p = \sum_{i=1}^n x_i^p$ ,  $1 \leq p \leq d$ , which is the sum of data values of  $p$ -th attribute;  $SST = \sum_{i=1}^n (T_i)^2$  is the sum of the squares of timestamp  $T_1 \dots T_n$ ;  $ST = \sum_{i=1}^n T_i$  is the sum of timestamp  $T_1 \dots T_n$ ; while  $n$  is the number of data points.

Figure 2 shows the micro-cluster generation process with parameters defined below:

- $P = \{x_i \mid 1 \leq i \leq n\}$  represents the data after perturbation.
- $Q$  represents the micro-cluster number set by user.
- $M$  represents the input micro-cluster.
- $q$  represents the current number of cluster centers,  $0 \leq q \leq Q$ .

- $x_i$  and  $x_j$  ( $1 \leq i \leq n, 1 \leq j \leq n, i \neq j$ ) represent two randomly selected data points,  $n$  is the data point number.
- $d^2(x_i, x_j)$  represents the squared Euclidean distance between  $x_i$  and  $x_j$ .
- $D = \{d^2(x_i, x_j) \mid 1 \leq i \leq n, 1 \leq j \leq n, i \neq j\}$  represents the set of the squared Euclidean distance between any two arbitrary data points, with an initial value of  $\phi$ .
- $s_k$  represents a cluster center,  $1 \leq k \leq q$ .
- $S = \{s_k \mid 1 \leq k \leq q\}$  represents the set of current cluster center, with an initial value of  $\phi$ .
- $d^2(x_i, s_k)$  represents the squared Euclidean distance between  $x_i$  and  $s_k$ .
- $D_i = \{d^2(x_i, s_k) \mid 1 \leq i \leq n, 1 \leq k \leq q\}$  represents the set of the squared Euclidean distance between data point  $x_i$  and each of the cluster center, with an initial value of  $\phi$ .

<b>Input :</b> $P, Q$
<b>Output :</b> $M$
<pre> 1. For each <math>x_i, x_j</math> do <math>D \leftarrow D \cup \{d^2(x_i, x_j)\}</math>; 2. <math>S \leftarrow S \cup \{x, y\}</math> s.t. <math>d^2(x, y) = \text{Max}[D]</math>; <math>q \leftarrow q+2</math>; 3. While (<math>q &lt; Q</math>) do {3.1 For each <math>x_i \notin S</math> do   {3.1.1 For each <math>s_k \in S</math> do     <math>D_i \leftarrow D_i \cup \{d^2(x_i, s_k)\}</math>;   3.1.2 <math>D_{\min} \leftarrow D_{\min} \cup \{\text{Min}[D_i]\}</math>;   3.2 <math>S \leftarrow S \cup \{x_i\}</math> s.t. <math>\text{Min}[D_i] = \text{Max}[D_{\min}]</math>;     <math>q \leftarrow q+1</math>;} 4. Use K-means to generate <math>M</math>;</pre>

**Figure2. Micro-cluster generation process.**

Detailed steps are described below:

- Step 1: Calculate the squared Euclidean distance  $d^2(x_i, x_j)$  between each data point  $x_i$  and  $x_j$ .
- Step 2: Find the two data points that have the longest distance, store the distance to the cluster center set  $S$  and add the cluster center number  $q$  by 2.
- Step 3: Determine if the current cluster center number  $q$  equals to the micro-cluster number  $Q$  set by the user, if affirmative then execute Step 4, if not execute Step 3.1 and Step 3.2.
- Step3.1: Execute Step 3.1.1 and Step 3.1.2 on each data point  $x_i$  outside of the cluster center.
- Step 3.1.1: Calculate the squared Euclidean distance  $d^2(x_i, s_k)$  between the data point  $x_i$  and the cluster center  $s_k$  for each cluster center  $s_k$ .
- Step 3.1.2: Find the minimal value of the squared Euclidean distance between each data point  $x_i$  and each cluster center.
- Step 3.2: Find the maximal value of  $D_{\min}$ , set the data point  $x_i$  as the new cluster center. Then add the cluster center number  $q$  by 1, return to Step 3.

Step 4: Set  $Q$  number of cluster center as the initialized cluster center and generate  $Q$  number of micro-cluster  $M$  using K-means algorithm.

When a new stream data which has been rotated and perturbed incoming, calculate the squared Euclidean distance between each data point and each micro-cluster center (the cluster feature vector  $TS$  inside the micro-clusters is divided by  $n$ ) to find out the nearest micro-cluster from each data point. Then using cluster feature vector, determine if the new stream data is smaller than the maximal boundary value  $t$  of the nearest micro-cluster. The so-called maximal boundary value is the root mean square deviation from the data point inside the micro-clusters to the micro-cluster center, as shown on formula 3. If the data number of the nearest micro-cluster is 1, the maximal boundary value is set as  $\alpha$  times more than the root mean square deviation of the second nearest micro-cluster, with the  $\alpha$  value set by the user. When the new stream data is smaller than the maximal boundary value, then the new stream data should be absorbed by the existing micro-cluster and the statistical information inside the micro-cluster is updated, with the steps described below:

$$\begin{aligned}\sum_{i=1}^n (x_i^p)^2 &= \sum_{i=1}^n (x_i^p)^2 + (x_{i+1}^p)^2; \\ \sum_{i=1}^n x_i^p &= \sum_{i=1}^n x_i^p + x_{i+1}^p; \\ \sum_{i=1}^n (T_i)^2 &= \sum_{i=1}^n (T_i)^2 + (T_{i+1})^2; \\ \sum_{i=1}^n T_i &= \sum_{i=1}^n T_i + T_{i+1}; \\ n &= n + 1;\end{aligned}$$

If the new stream data is not smaller than the maximal boundary value, then establish a new micro-cluster.

When establishing a new micro-cluster, due to limited memory space, an existing micro cluster must be reduced in order to free a memory space, which is done through delete or join the existing micro-cluster to achieve this purpose. First check for the existence of any micro-cluster considered as outlier by estimating the average timestamp of the most recent data point  $m$  from each micro-cluster, delete the micro-cluster with the minimal average timestamp. However in a data stream environment, it is unlikely to store the most recent data point  $m$  of all micro-clusters. To solve this issue, assume timestamp as normal distribution and proceed with the following procedures. When the data quantity  $n$  inside the micro-cluster is smaller than  $2 \times m$ , directly use the timestamp of the micro-cluster to calculate the timestamp mean,  $ST/n$ , which is used as the average timestamp for the data point of each micro-cluster. Otherwise use the timestamp mean, standard deviation  $\sqrt{SST/n - (ST/n)^2}$  and the Z-score calculated from the timestamp data from the micro-cluster, and through formula 4 to estimate the average timestamp for  $m/(2 \times n)\%$  of the data point in each micro-cluster, thereby obtaining an

estimated value of recent stamp. If the smallest recent stamp of all micro clusters is smaller than the boundary value  $\delta$  defined by the user, then that particular micro-cluster should be deleted. If all recent stamps are greater than the boundary  $\delta$ , then combine the two nearest micro-clusters. Assume micro-cluster A and micro-cluster B are combined as micro-cluster AB, the statistical information for updating micro-cluster are described below:

$$\begin{aligned}(\sum_{i=1}^n (x_i^p)^2)^{AB} &= (\sum_{i=1}^n (x_i^p)^2)^A + (\sum_{i=1}^n (x_i^p)^2)^B; \\ (\sum_{i=1}^n x_i^p)^{AB} &= (\sum_{i=1}^n x_i^p)^A + (\sum_{i=1}^n x_i^p)^B; \\ (\sum_{i=1}^n (T_i)^2)^{AB} &= (\sum_{i=1}^n (T_i)^2)^A + (\sum_{i=1}^n (T_i)^2)^B; \\ (\sum_{i=1}^n T_i)^{AB} &= (\sum_{i=1}^n T_i)^A + (\sum_{i=1}^n T_i)^B; \\ n^{AB} &= n^A + n^B;\end{aligned}$$

The combined micro-cluster  $id$  is the union of the  $id$  for both micro-clusters.

### 2.2.2 Geometric Time Frame Allocation

The updated micro-clusters are stored using geometric time frame allocation through snapshot form. In comparison with the traditional pyramidal time frame, geometric time frame has solved the redundancy resulted from pyramidal time frame, enhancing more efficiency for memory use. Geometric time frame allocates snapshots to different frame numbers, with the number lying between 0 and  $\log_2(T)$ , with  $T$  referring to the longest time length of data stream, while the allocated frame numbers for snapshots refer to the degree of granularity for the stored snapshot. The snapshots stored in frame number  $i$  whose moment must meet the condition of divisible by  $2^i$ , therefore the snapshots stored in frame number 0 will have odd-numbered moments. In addition, assume  $max\_capacity$  is the maximum stored snapshots for each level, and the limit for the maximum frame number should not exceed  $\log_2(T)$  from the previous information, and from here we know that the maximum snapshot numbers stored starting from data stream to time unit  $T$  is  $(max\_capacity) \times \log_2(T)$ . The proceeding is the principle for snapshots of geometric time frame allocation: Assume  $s$  is the new snapshot, when  $s$  enters the geometric time frame, it is required to determine if  $s$  is divisible by  $2^i$ , and if  $s$  is divisible by  $2^i$  and not divisible by  $2^{i+1}$ , then  $s$  is inserted into the level of frame number  $i$ . Due to each level contains a maximum storage quantity, if assuming level  $i$  has reached its maximum storage quantity, then the snapshot of the earliest moment of that level will be removed and stored into storage and inserted with the snapshot of the latest moment.

### 2.2.3 Macro-Cluster Generation

Macro-cluster generation has become a process for re-clustering on stored micro-clusters with reference on user demand. Due to micro-cluster reflects the overall time

information since the start of data streams, therefore the subtractive characteristic of feature vector is used according to the micro-cluster  $id$  to find out the time scope of micro-clusters set by user. Assume the current time is  $t_c$ , users would like to mine on the data during the period  $h$  from current to period of past experiences in order to obtain  $K$  clustering result. Under the condition given, we will need to find the snapshots stored before time  $t_c-h$ . We take  $S(t_c-h)$  to represent the micro-cluster set for time  $t_c-h$ , take  $S(t_c)$  to represent the micro-cluster set for time  $t_c$ , whereas  $h$  refers to the tolerance for error for time  $t_c-h$  previously set by user. For each micro-cluster in  $S(t_c)$ , find out the micro-cluster that conform to  $S(t_c-h)$  according to its individual  $id$ , and reduce the cluster feature vector what conforms to the micro-cluster of  $S(t_c-h)$ . This approach will ensure the micro-cluster generated during the period  $h$  set by user will not influence the mining result. Then, use the micro-cluster center as the macro-point in conformity with the period  $h$  for user observation, then take the data point quantity contained in the macro-point as weight to select  $K$  number of the data points as the cluster center for macro-clustering, using macro-cluster generation process shown in Figure 3 to cluster for generation of  $K$  number of macro-clusters, with parameters definitions described below:

- $W = \{m_i \mid 1 \leq i \leq Q\}$  represents the macro-point set for the observation period  $h$  for user.
- $C = \{s_j \mid 1 \leq j \leq K\}$  represents the set of current cluster centers, with the initial value selected from the  $K$  number with the largest value according to the data number  $n$  of each macro-point  $m_i$ .
- $K$  represents the macro-cluster number set by the user.
- $G_j$  represents a macro-cluster,  $1 \leq j \leq K$ .
- $m_i$  represents a macro-point,  $1 \leq i \leq Q$ .
- $n_i$  represents the data number in macro-point  $m_i$ ,  $1 \leq i \leq Q$ .
- $s_j$  represents a cluster center,  $1 \leq j \leq K$ .
- $d^2(m_i, s_j)$  represents the squared Euclidean distance between  $m_i$  and  $s_j$ .
- $D_i = \{d^2(m_i, s_j) \mid 1 \leq j \leq K\}$  represents the set of the squared Euclidean distance between macro-point  $m_i$  and each cluster center  $s_j$ .
- $Pointdis[i]$  is used for storing the Squared Euclidean Distance between macro-point  $m_i$  and the nearest cluster center,  $1 \leq i \leq l$ .
- $CenterM[i]$  is used for storing the current cluster center for macro-point  $m_i$ ,  $1 \leq i \leq l$ .
- $d^2(m_i, CenterM[i])$  represents the squared Euclidean distance between the macro-point  $m_i$  and the current cluster center  $CenterM[i]$ .

Detailed steps are described below:

- Step 1: Calculate the squared Euclidean distance  $d^2(m_i, s_j)$  between each macro-point  $m_i$  and each cluster center  $s_j$ .
- Step 2: Find the minimal value of the squared Euclidean

distance between each macro-point  $m_i$  with each cluster center  $s_j$ , store the value to  $Pointdis[i]$  while store the current cluster center  $s_j$  to  $CenterM[i]$ .

<p><b>Input :</b> <math>W, C, K</math></p> <p><b>Output :</b> <math>C, G_j</math></p>
<pre> 1. For each <math>m_i, s_j</math> do <math>D_i \leftarrow D_i \cup \{d^2(m_i, s_j)\}</math>; 2. For each <math>m_i</math> do <math>Pointdis[i] \leftarrow \text{Min}[D_i]</math>;    <math>CenterM[i] \leftarrow s_j</math> s.t. <math>d^2(m_i, s_j) = \text{Min}[D_i]</math>; 3. For each <math>s_j</math> do    <math>s_j \leftarrow \frac{\sum n_i \times m_i}{\sum n_i}</math> s.t. <math>CenterM[i] = s_j</math>; 4. For each <math>m_i, s_j</math> do <math>D_i \leftarrow D_i \cup \{d^2(m_i, s_j)\}</math>; 5. For each <math>m_i</math> do    <math>CenterM[i] \leftarrow s_j</math> s.t. <math>s_j d^2(m_i, s_j) = \text{Min}[D_i]</math>; 6. While <math>\exists i [d^2(m_i, CenterM[i]) &gt; Pointdis[i]]</math> do   {6.1 For each <math>m_i</math> s.t. <math>d^2(m_i, CenterM[i]) &gt; Pointdis[i]</math> do   {6.1.1 For each <math>m_i, s_j</math> do <math>D_i \leftarrow D_i \cup \{d^2(m_i, s_j)\}</math>;   6.1.2 For each <math>m_i</math> do <math>Pointdis[i] \leftarrow \text{Min}[D_i]</math>;    <math>CenterM[i] \leftarrow s_j</math> s.t. <math>d^2(m_i, s_j) = \text{Min}[D_i]</math>;   6.1.3 For each <math>s_j</math> do    <math>s_j \leftarrow \frac{\sum n_i \times m_i}{\sum n_i}</math> s.t. <math>CenterM[i] = s_j</math>;   6.1.4 For each <math>m_i, s_j</math> do <math>D_i \leftarrow D_i \cup \{d^2(m_i, s_j)\}</math>;   6.1.5 For each <math>m_i</math> do    <math>CenterM[i] \leftarrow s_j</math> s.t. <math>s_j d^2(m_i, s_j) = \text{Min}[D_i]</math>;}} 7. For each <math>m_i</math> do    <math>G_j \leftarrow G_j \cup \{m_i\}</math> s.t. <math>CenterM[i] = s_j</math>; </pre>

**Figure 3. Marco-cluster generation process.**

- Step 3: For each current cluster center  $s_j$ , calculate the weighted mean inside the cluster and store the result to  $s_j$ .
- Step 4: For each macro-point  $m_i$ , recalculate the squared Euclidean distance  $d^2(m_i, s_j)$  between the macro-point and each cluster center  $s_j$ .
- Step 5: For each macro-point  $m_i$ , store the current cluster center  $s_j$  to  $CenterM[i]$ .
- Step 6: Determine if the distance  $d^2(m_i, CenterM[i])$  between any arbitrary macro-point  $m_i$  and the current cluster center is greater than the distance stores for  $m_i$  stored in  $Pointdis[i]$ , if affirmative then execute Step 6.1, or else execute Step 7.
- Step 6.1: For the squared Euclidean distance  $d^2(m_i, CenterM[i])$  between the current clusters is greater than the distance  $Pointdis[i]$  stored at each macro-point  $m_i$ , execute Step 6.1.1 to Step 6.1.5.
- Step 6.1.1: For each macro-point  $m_i$ , recalculate the squared Euclidean distance  $d^2(m_i, s_j)$  between the macro-point  $m_i$  and each cluster

center  $s_j$ .

Step 6.1.2: Find out the minimal value of the squared Euclidean distance between each macro-point  $m_i$  and each cluster center  $s_j$ , then store the value to  $Pointdis[i]$  and store the current cluster center  $s_j$  to  $CenterM[i]$ .

Step 6.1.3: Calculate the weighted mean of the cluster for each current cluster center  $s_j$  and store the result to  $s_j$ .

Step 6.1.4: Recalculate the squared Euclidean distance  $d^2(m_i, s_j)$  between each macro-point  $m_i$  and each cluster center  $s_j$ .

Step 6.1.5: Store the current cluster center  $s_j$  for each macro-point  $m_i$  to  $CenterM[i]$ .

Step 7: Store each macro-point  $m_i$  to its belonging macro-cluster  $G_j$ .

### 3. Performance Evaluation

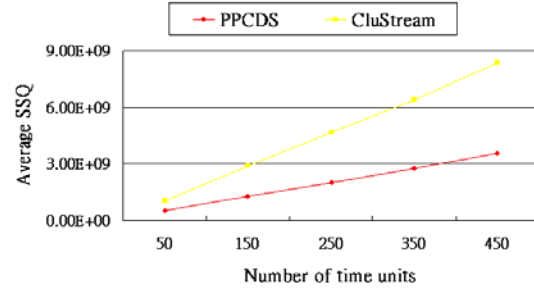
#### 3.1 Accuracy Evaluation of PPCDS

In the experiment of accuracy evaluation, we use the average of the sum of square distance, also known as the Average SSQ to evaluate accuracy, whereas the smaller the value of the Average SSQ, means the higher the accuracy. The data source is the real datasets KDD-CUP'98, with experimental parameters set to  $n = 2000$  and  $t = 2$ . The so-called Average SSQ consists of the following definitions: Assume there are  $W$  number of macro-points in the period  $h$  before the current moment  $T_c$ , find the cluster center with the nearest distance to each macro-point  $m_i$  and calculate the squared Euclidean distance  $d^2(m_i, s_j)$  between  $m_i$  and  $s_j$ , consequently the Average SSQ of period  $h$  before the current moment  $T_c$  is equal to the total sum of the squared Euclidean distance between the cluster centers and all  $W$  number of macro-points in period  $h$ , divided by the macro-cluster number  $K$ , as shown on formula 5. Figure 4 shows the circumstance of changes in mining accuracy in different period  $h$  and data stream rate SP, with  $SP = 200$  referring to data streams flow in at the rate of 200 data points per every time unit. The horizontal axis represents a different time unit quantity, while the vertical axis represents the Average SSQ. It is noted from the figure that despite the data must undergo a privacy-preserving treatment through rotation perturbation during the mining process, nonetheless due to rotation perturbation contains characteristics of isometric transformation, and consequently it will not cause much impact on the accuracy of mining results. In addition, in the micro-cluster generation Process, the quality micro-cluster generated through optimization of cluster center will further enhance the mining accuracy.

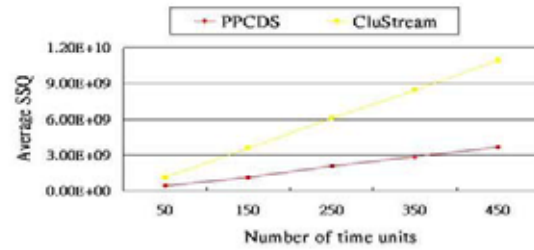
#### 3.2 Scalability Evaluation of PPCDS

In the experiment of scalability evaluation, the primary test emphasizes on the data stream processing capability of PPCDS, with data sources from real datasets, and experimental parameters set to  $n = 2000$ ,  $t = 2$  and  $SP = 2000$  re-

spectively. Figure 5 shows the processing capability of PPCDS on stream data, with the horizontal axis referring to the elapsed time in unit of second to data processing, while the vertical axis referring to the data point quantity processed in each second. As shown in the figure, due to PPCDS start performing rotation perturbation on data and establishing micro-cluster with incoming stream data. Consequently it causes a poor efficiency on the initial data processing, with the time approximately at 20 seconds. Due to the generation of micro-clusters allows the data undergoing rotation perturbation treatment to directly cluster stream data, gradually stabilizing process efficiency.



(a)  $h = 4$ ,  $SP = 200$



(b)  $h = 16$ ,  $SP = 200$

Figure 4. Comparison of mining accuracy.

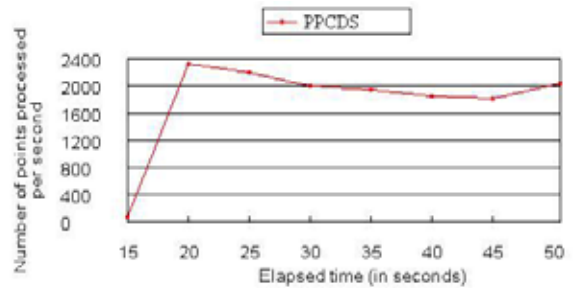


Figure 5. Stream data processing efficiency.

Furthermore, through setting different number of dimensions and number of cluster, we observe the time required for PPCDS in stream data processing. In the experiment of testing the impact of number of dimensions on scalability, we use three artificial datasets of B400C20 (representing 400K data points and 20 clusters), B200C10 (representing 200K data points and 10 clusters) and B100C5 (representing 100K data points and 5 clusters) respectively, with

number of dimensions variation from 10 to 80. Figure 6 shows the execution time of PPCDS in different number of dimensions, with the horizontal axis indicating the different number of dimensions and the vertical axis indicating the execution time in units of seconds.

In the experiment of the impact of testing the number of cluster on scalability, similarly we use three artificial datasets of B400D40 (representing 400K data points and 40 dimensions), B200D20 (representing 200K data points and 20 dimensions) and B100D10 (representing 100K data points and 10 dimensions), with the variation of number of clusters from 5 to 40. Figure 7 shows the variation of execution time for PPCDS in different number of cluster, with the horizontal axis indicating the different number of clusters, while the vertical axis indicating the execution time in units of seconds.

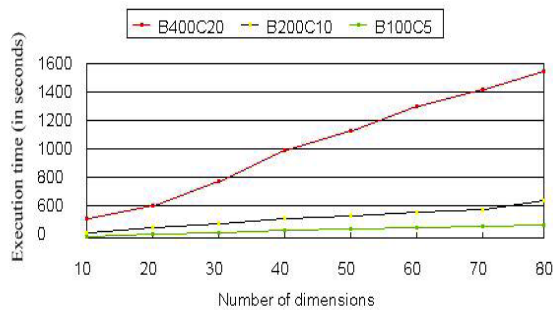


Figure 6. Impact of variation on number of dimensions.

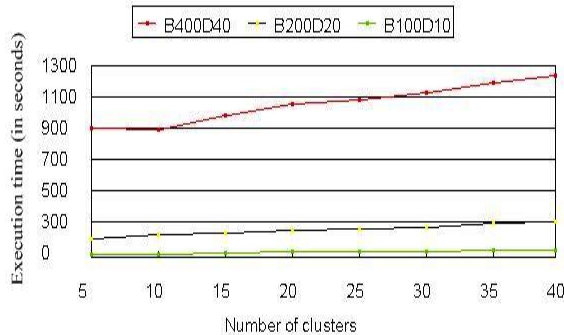


Figure 7. Impact of variation on number of clusters.

### 3.3 Sensitivity Evaluation of PPCDS

In order to obtain a high accuracy mining result, the number of micro-cluster must far exceed the number of macro-cluster, however excessive micro-cluster will reduce the execution efficiency and the memory use benefits, therefore how to strike a balance between mining accuracy and storage benefits becomes relatively significant. In this experiment, we use KDD-CUP'98 datasets as the data source, and through controlling the number of micro-cluster, using micro-cluster ratio and the Average SSQ to evaluate the impacts of the number of micro-cluster on mining accuracy. The so-called micro-cluster ratio refers to the number of micro-cluster divided by the number of macro-cluster.

Figure 8 shows the impact of micro-cluster ration on accuracy, with the horizontal axis indicating different micro-cluster ratio, while the vertical axis indicating the Average SSQ. We fix the number of time unit as 200,  $SP = 200$  and  $h = 16$ . It is noted from the figure, if the number of micro-cluster used equals to the number of macro-cluster, then we will obtain a poor mining accuracy, mainly resulted from the number of micro-cluster used is too small. However when the micro-cluster ratio increases, the mining accuracy will increase accordingly. When the micro-cluster ratio increases to 15 approximately, the mining accuracy will turn to stabilized. The result indicates that it is not required to set the number of micro-cluster with a large number to obtain a good mining accuracy, provided that the numbers of micro-cluster and macro-cluster reach to a certain ratio.

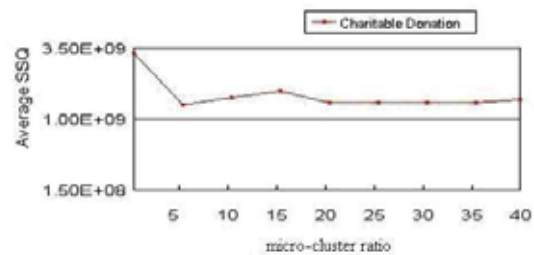


Figure 8. Sensitivity analysis on micro-cluster.

## 4. Conclusion

This paper proposes PPCDS for privacy-preserving clustering of data streams. Experimental results show PPCDS only requires a reasonable amount of memory to obtain a good mining accuracy. Besides, PPCDS not only can preserve data privacy but also can efficiently and accurately mine data streams.

## Acknowledgment

The authors would like to express their appreciation for the financial support from the National Science Council of Republic of China under Project No. NSC 96-2221-E-031-001-MY2.

## References

- [1] Agrawal, R. and Srikant, R., "Privacy-Preserving Data Mining," Proc. of 2000 ACM International Conference on Management of Data, Dallas, Texas, U.S.A., pp. 439-450 (2000).
- [2] Ordonez, C., "Clustering Binary Data Streams with K-means," Proc. of 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego, California, pp. 12-19 (2003).
- [3] Aggarwal, C., Han, J., Wang, J. and Yu, P. S., "A Framework for Clustering Evolving Data Streams," Proc. of 29th International Conference on VLDB, Berlin, Germany, pp. 81-92 (2003).
- [4] Yang, C. and Zhou, J., "HClustream: a Novel Approach for Clustering Evolving Heterogeneous Data Stream," Proc. of 6th IEEE International Conference on Data Mining, Hong Kong, China, pp. 682-688 (2006).



# FiGD: An Open Source Intellectual Property Violation Detector

Carson Brown, David Barrera, Dwight Deugo  
The School of Computer Science, Carleton University  
Ottawa, Ontario, Canada

carson@ccsl.carleton.ca, dbarrera@ccsl.carleton.ca, deugo@scs.carleton.ca

**Abstract:** *FiGD (Fingerprint Generator/Detector) is an open source Java application capable of detecting intellectual property violations in compiled Java programs without requiring access to the original source files. FiGD uses a modification of the  $n$ -gram method which is very accurate in discovering everything from blatantly copied source, to more advanced attempts of obfuscation (such as variable refactoring or white-space insertions). Our improvements to the algorithm allow us to increase the speed of detection and create small fingerprints which can be stored for future comparisons.*

## 1. Introduction

In recent years, Open Source Software (OSS) has seen a surge in popularity. It is now common to find OSS running on a variety of systems ranging from web servers [14] to super-computers to mobile phones. There are currently numerous OSS projects which have reached a level of maturity sufficient for use by governments and large corporations [15]. As this software makes its way into more areas, legal concerns begin to emerge. It is unclear who is at fault when an open source library in a commercial product fails. Open source licenses [12] can also be incompatible with each other, creating legal problems for companies developing OSS. These are legitimate concerns, but they are difficult to address if the origins of the code are unknown.

### 1.1 Problem

The problem we focus on in this paper is clone detection for software. We define a software clone as source code in a unknown project that has been copied (either fully or in part) from a known project. Clone detection is useful for pinpointing code theft, as well for general code auditing. Although clone detection has already been extensively researched, this paper focuses only on a small part of the problem which applies to software written in the Java programming language. We assume a black-box (A device or system whose workings are not understood by, nor accessible to, the user and is thus viewed in terms of its input and output characteristics) approach where we generally do not have access to the source code of the projects we are analyzing.

## 1.2 Motivation

The main motivation of this paper is to contribute to the Open Source philosophy. When open source software is stolen, any changes, improvements or otherwise, made by the intellectual property (IP) thief are unlikely to make it back into the community. Since the open source software development cycle relies heavily on developers contributing, IP theft can prove to be a dangerous threat to this particular ecosystem.

Another motivation is cost: we would like to make it affordable for companies or developers to audit their code for the existence of other OSS. As of October 2008, there are no known open source tools that (easily) allow this. Some available software packages allow source code comparison through simple string-matching, and others are designed to work with only specific programming languages. There are two commercial solutions, [13, 10], costing between \$50,000 to \$250,000 for annual subscriptions. Both of these companies also allow the end user to pay by the megabyte (Mb), but still at prices ranging from \$3,000 to \$25,000 for less than 100 Mb. Prices this high could prove to be a significant barrier of entry for small and medium-sized businesses.

## 1.3 Goals

Our goal is to write an application that will have the following functions:

- Generate a unique signature (fingerprint) from a Java ARchive (JAR) file.
- Search for similarities between a previously generated fingerprint and a new, unknown JAR file.
- Output relevant information regarding the matches found and percentage of certainty.

## 1.4 Objectives

Given the goals described in Section 1.3, our objectives are to focus not only on accuracy, but also on performance and system resources. The current string-matching approaches found in other projects [3, 4, 16] tend to be very precise but extremely slow, on the order of  $O(n^2)$ . These approaches also assume access to the

original source code which is not always provided. Our objective is also to avoid using large amounts of memory while generating our fingerprints or performing a comparison. These concerns are of particular importance when fingerprinting large files (i.e., >5Mb).

As another improvement, we will also avoid looking at source code. We believe that since the source code is not always packaged within JAR files, it would be better to work without relying upon it, and base our comparison on compiled Java byte-codes (Java byte-codes are what the Java Virtual Machine (JVM) actually executes. It is the compiled version of source code, where each byte-code instruction is exactly one byte in length [7].).

The final objective is to make our fingerprint-detector immune to variable refactoring (to change all references to a variable, for example, to a different name). It is in this manner that our algorithm will still detect a match based on functionality, but not on semantics. Our algorithm is thus resistant to changing variable names, method names, or class names.

In order to generate small fingerprints, we will find and store parts of the JAR file which are highly representative of that file only. In essence, this technique closely resembles what is done in the anti-virus industry (and, in fact, in any signature-based detection environment) where the smallest matching string of a virus is used as a signature. Anti-virus software programs are able to rapidly look for thousands of signatures in a given file. We have created software that achieves similar behaviour at the Java method level.

## 1.5 Outline

The remainder of the paper is structured as follows: Section 2 describes certain basic concepts and terminology, as well as look at related projects that attempt to solve a similar problem. Section 3 explains our design strategy, including decisions that had to be made in order to reach our objectives. In Section 4 we present our results. Section 5 provides a conclusion and elaborates on future work.

## 2. BACKGROUND

Clone detection can usually be done either by string-matching the source code or looking at binary file signatures. The string-matching technique requires looking at small sub-strings in the file (called *n-grams*, where *n* is the number of characters in the string, or gram) and then try to identify those strings in a different file. This obviously requires a large amount of memory and processing, especially if a sliding window of the entire file is taken. For example, if the original file has 1000 characters in total (including white spaces and line termination), using *n-grams* of size 10 without a sliding window would give us 100 10-grams. If a sliding window were used, we would have 991 10-grams (1000-10+1). Continuing this example, we would need to search for occurrences of 991 strings in a new file.

Binary file signature matching provides the added benefit that the original source code is not required. This is useful, for example, in the anti-virus industry, where viruses and worms are packaged and distributed globally. There is a slight difference that prevents us from using this approach directly: source code may be slightly modified and rebuilt, producing a completely different binary file. For example we take the (extremely simple) method in Listing 1:

```
int method() {
    int i=10;
    return i;
}
```

Listing 1: Simple Method

This method would have a certain binary signature once compiled. However if we were to change its source to the following, Listing 2:

```
int method() {
    int i=10;
    i--;
    i++;
    return i;
}
```

Listing 2: Modified Simple Method

The binary signature may be completely different, even though the method has no changes in functionality (i.e., it still returns the value of *i=10*). This problem is of serious concern when considering OSS fingerprinting, as the source code is almost always easily available and can be changed and compiled by any software recipient.

### 2.1 JAR files and Class files

Our software will take as input any valid JAR file [5]. A JAR file is a file-type based on the popular “ZIP” file format. It was developed by Sun Microsystems, and it allows many files to be aggregated into one, with optional compression. JAR files contain the Java resources necessary to run Java programs. For this paper, we are interested in one set of resources called “Class” files [8].

Class files are Java’s compiled files. A source file (usually ending in .java) will be compiled to produce one or more class files which are (for the most part) platform independent (excluding platform-specific system functions). Class files contain byte-code groups of Java’s instruction sets that will be run (or interpreted) inside a JVM.

### 2.2 Related Software

Although there are countless papers on clone detection [2], software products that can detect clones of compiled Java programs are difficult to find. Many papers describe early prototypes of their algorithms and therefore have not yet released their software. Other papers describe

the best ways of comparing strings, but generally require access to source files. Software such as Simian [6], Clone Digger [3], CCFinderX [1] and Clone Doctor [4] are readily available, but also work only source files. The advantage of these tools, however, is that they should work on any type of source code (C, Python, Java, assembly, even plain text) since they are performing basic string matching techniques.

### 3. APPROACH

In this section describes at a high level our approach. We provide the main algorithms for FiGD as well as what decisions had to be made in order to achieve our goals.

#### 3.1 Design

Rather than create a general purpose fingerprinting program, this project has the particular distinction of comparing JAR files. These have a known composition, both in compression and file structure. For the purposes of this project, we are looking for code reuse from one JAR to another. Thus, we have distilled our approach from the general case considerably: our fingerprint generator and detector considers only Java class files, and more specific still, considers only the byte codes of each method contained in these class files.

We have made this decision based on what we feel is representative of the uniqueness of a JAR file. When considering JAR files, we cannot guarantee the inclusion of source code (Java or otherwise), nor can we guarantee that any part of the comparison JAR file retains similar naming or folder structure of Java packages. What we can consider, however, is that the essence of a Java method will be retained, regardless of moving the method to another class or changing its name. That is to say, the method will still do the same thing.

This section has been broken down into two sections, comparing the two components of the project: the fingerprint generator, and the detector.

##### 3.1.1 Fingerprint Generator

The fingerprint generator must first open the JAR file to be compared. All JAR files are created with the ZIP standard, and can be decompressed rather easily. In the Java API, the `java.util.jar` package contains many useful objects, including the `JarFile` and `JarEntry` classes. It is then possible to compute a listing of all files contained in the `JarFile` object, and a simple file type check allows for a complete listing of all class files.

The decompressing of the JAR file contents into class files is done through the `JarResources` class, adapted from a Java-World article [9]. We have modified the class to only decompress the JAR's class files into memory. Our fingerprint generator can then iterate over all class files, by requesting each class file individually from the `JarResources` object. This is done by writing the class out to a temporary file, which is later deleted upon the

program's exit. FiGD incurs in a slight memory overhead due to the extensive utilization of objects as opposed to programming in a structural language such as C. This, however, proves to be a negligible performance limitation, since the JAR files we are testing usually fall within the 0Mb-50Mb file size range.

With the Java class file written out to a temporary file, we made use of another open source library to access the necessary methods. The `org.netbeans.modules.classfile` package [11] allows for direct access to the class file byte-codes, by loading the file as a `ClassFile` object, part of the NetBeans package. It is then possible to iterate through all methods of the `ClassFile` object, which are available as instances of the `Method` class. Each method can then be extracted as a list of Java byte-codes using other classes found in the NetBeans package.

Rather than use the byte-codes for a whole method (which would increase the size of our fingerprint considerably), we decided on only storing a single  $n$ -gram per method. We first compute all  $n$ -grams of each method, then the most unique  $n$ -gram is selected to represent that method in the JAR file's fingerprint. The uniqueness of  $n$ -grams differ based on the size of  $n$ , but our testing has shown that using  $n$  a gram size of 10 (i.e., 10 byte-codes) strikes a good balance between accuracy and fingerprint size. Also, using larger values for  $n$  did not improve accuracy. By using this approach, the fingerprint size is linearly dependent on the number of methods found in the JAR file. It is this list of unique  $n$ -grams, as well as some statistical information—such as the number of methods,  $n$ -grams stored and total  $n$ -grams—that form the fingerprint of a JAR file.

##### 3.1.2 Detector

Detection requires an original fingerprint as well as a comparison JAR file. The result returned from our detector contains both our certainty percentage that code from the fingerprinted JAR file is contained in the comparison, and also our calculation of how much of that original code appears. This is calculated by opening the JAR in much the same way as the fingerprint generator, save that our generator does not throw away non-unique  $n$ -grams but instead compares these to the representative  $n$ -grams of the fingerprint. This is done by first generating a list of  $n$ -grams for a given method in a class file. These are then compared to the  $n$ -grams in the fingerprint which have not already been matched by  $n$ -grams in the comparison JAR file. The list of  $n$ -grams generated by the detector are not stored for later use: the only  $n$ -grams stored in working memory are those that are being compared to the fingerprint. When a match has been found between the fingerprint and the comparison JAR (i.e.: both JAR files contain the same method) the next method in the comparison JAR is considered for detection.

The number of matches is stored, and used in the calculations of the detector's final result. The number of matches divided by the total number of  $n$ -grams in the

original fingerprint yields the percentage of the original JAR file in the comparison JAR file. The certainty of the final result is calculated by the percentage of the  $n$ -grams included in the original fingerprint divided by the total number of  $n$ -grams created from the JAR file.

### 3.1.3 Summary

The algorithms in Listing 3 and 4 detail the fingerprint generation and detection approaches from Sections 3.1.1 and 3.1.2.

Algorithm Generator

Input: Jar File

Output: Fingerprint

1.  $G \leftarrow \{ \emptyset \}$
2.  $C \leftarrow \{ c \mid \forall \text{ Class File } c \in \text{Jar File} \}$
3. for  $c \in C$
4.     do  $M \leftarrow \{ m \mid \forall \text{ Method } m \in c \}$
5.         for  $m \in M$
6.             do compute  $n$ -grams  
               from byte-codes of  $m$
7.                  $s \leftarrow n$ -gram of lowest count
8.             add  $s$  to  $G$
9. add  $G$  to Fingerprint
10. return Fingerprint

Listing 3: Generator Algorithm

Algorithm Detector

Input: Fingerprint

Input: Jar File

Output: FingerprintResult

1. count  $\leftarrow 0$
2.  $C \leftarrow \{ c \mid \forall \text{ Class File } c \in \text{Jar File} \}$
3. for  $c \in C$
4.     do  $M \leftarrow \{ m \mid \forall \text{ Method } m \in c \}$
5.         for  $m \in M$
6.             do consider each  $n$ -gram  $g_c$  of  $m$ :
7.                 if  $g_c \in \text{Fingerprint}$
8.                     then count  $\leftarrow$  count + 1
9.             remove  $g_c$  from considered  
                   Fingerprint entries
10.             continue to next Method  $m$
11. certainty  $\leftarrow$  count / Fingerprint<sub>size</sub> \* 100%
12. add certainty to FingerprintResult
13. return FingerprintResult

Listing 4: Detector Algorithm

## 3.2 Decisions Made

Over the course of creating the fingerprint generator and detector, we made a variety of design decisions. Our first implementation for creating fingerprints at the Java method level involved computing simple hashes of every

method, which significantly reduced our accuracy when situations such those described in Section 2. This inability to catch “useless” modifications to the code in a method led us to desire a way of capturing the uniqueness of a method. We then implemented the  $n$ -gram implementation, and extracted only the first  $n$ -gram of lowest frequency. This involved some loss in accuracy, but it is our belief—proven through testing—that this loss is negligible when compared to the large decrease of the generated fingerprint’s footprint.

Our experiments also show that using  $n$ -grams where  $n$  is 10 have shown to be the most representative. When  $n$  is set to lower values, accuracy of the algorithm suffers, as the  $n$ -grams represent very little of a method’s structure. This loss of accuracy can be attributed to false positives when comparing the fingerprint to another JAR file. Similar to the method hashing described above, having large values of  $n$  leads to loss of accuracy, where truly equivalent methods are no longer detected as such. As the size of  $n$  increases, the algorithm approaches behaviour similar to the method hashing described above.

Our implementation of generating fingerprints and detecting similarities between JAR files compares based on the contents of class file methods. This means that “empty” or unimplemented methods are not considered. We are aware that our implementation cannot properly deal with interface classes, or the non-implemented abstract methods found in abstract classes. We do not believe this to be a fault in our design, as interfaces are by definition public, and abstract classes are still considered; only the abstract methods are ignored.

## 4. RESULTS

This section documents the results of testing FiGD on various, representative JAR files. The subsections below describe testing in both accuracy and performance during the implementation’s construction and as a completed product in “real world” use cases.

### 4.1 Accuracy

For testing the accuracy of FiGD, we used two random JAR files found in the Eclipse JAVA IDE installation. We created fingerprints for each one, and then compared them to themselves using the detector. Both fingerprints were generated in under 3 seconds, and the output claiming a 100% match was displayed immediately after. A 99.999% certainty was also displayed in both cases, confirming that with high confidence, the files are fully identical.

One of the features of FiGD is that as soon as the first  $n$ -gram is matched for a given method, no further  $n$ -grams are compared for that method, since we assume we have found a cloned code segment. This greatly speeds up the detection phase when we know a priori that there is some kind of similarity between two files. If the files are completely different (i.e., zero matching methods), then our detector has to compare every single  $n$ -gram to the

fingerprint, which takes time  $O(n \cdot m)$ , where  $n$  and  $m$  represent the number of methods in each JAR file.

Although false positives have not been extensively tested, we believe the chance of them occurring is small, since our  $n$ -grams are large enough to make each method signature reasonably unique.

## 4.2 Performance

For performance testing, we used a large JAR file (about 10Mb) and a small JAR file (about 300Kb). We saved copies of both JAR files with slight modifications. The modifications were simply to remove a random number of class files from each one. We then compared the original unmodified file to the modified variations. We obtained results in less than 5 seconds with FiGD reporting between 70% and 80% matches between the JAR files. This seems correct, as only a small number of Class files were removed. The certainty percentage reported was still high at over 80% for both test cases, confirming that our algorithm is not only fast, but correct as well.

## 4.3 Real World Testing Results

Our various tests over the course of developing FiGD, made use of a variety of JAR files, including many from the Eclipse 3.4 Classic IDE plugin directory, chosen due to Eclipse’s popularity. Two JAR files have been included in Table 2 from this software: `org.eclipse.jface.text.3.4.0.v200806032000.jar` and `org.eclipse.jdt.ui.3.4.0.v20080603-2000.jar` (abbreviated in the table due to file name length). The third JAR file used, `commons-attributes-api-2.2.jar`, is from the Apache Commons library. These three JAR files include mcompiled Java class files and are three representative sizes, the largest being included for “stress” testing. The tests have been performed on a Toshiba Satellite PSM40-SF300E laptop, with an Intel Pentium M processor (1.86GHz, 533MHz FSB, 32KB of L1 cache, 2MB of L2 cache), 1 GB of memory (2 x 512 PC2700 DDR SODIMM) running Ubuntu 8.10 GNU/Linux.

The JAR files used in Table 1 have been created especially for testing FiGD, and include small, easy to manage Java class files used in first-year programming assignments. These class files have been modified and compiled into various JAR files, as described in the table. “Original” is in reference to an original set of Java class files serving a particular purpose. For each of the test cases where files were modified, a significant number of changes were made—for example, more than 60% of all variable names were changed for the second test in Table 1. These tests show that FiGD is insensitive to aesthetic source-code changes such as variable name refactoring or source code comments.

Table 2 demonstrates more “real world” testing, involving real world JAR files. These files were deliberately chosen because they are not obviously related by purpose or content. The certainty percentages

calculated are entirely dependent on how well FiGD can form a representative fingerprint on a given JAR file, while the inclusion percentage (Inc %) relies on the number of matched methods. These tests confirm our suspicions: that the JAR files are convincingly different. The only non-obvious data set is the last pair of tests comparing the two Eclipse-based JAR files. We believe these inclusion percentages to be correct, as both of these JAR files share a common Eclipse plug-in architecture, and likely do share similar code bases in this respect. These tests also confirm the worst-case running time calculated above, as these files have very few similarities, causing near quadratic run-times, executing over a few minutes on the test machine.

Table 1: Accuracy Testing

Description	Certainty	Inc (%)
Original compared to copy where method names were changed	100%	100
Original compared to copy where variables were renamed and comments added or removed	100%	100
Original compared to copy where additional class files were added	100%	100
Previous test in reverse	100%	61.6
Original compared to copy with methods and class files removed	100%	83.8
Above text in reverse	100%	100

Table 2: Performance Testing

Description	Certainty	Inc (%)	Time (ms)
<code>commons-attributes-api-2.2.jar</code> (35.9Kb) compared to itself	93.6%	100	220
<code>org.eclipse.jface.text.jar</code> (922.7Kb) compared to itself	88.5%	99.9	2137
<code>org.eclipse.jdt.ui.jar</code> (9.2Mb) compared to itself	97.6%	99.9	19304
<code>commons-attributes-api-2.2.jar</code> compared to <code>org.eclipse.jface.text.jar</code>	93.6%	0	3333
Previous test in reverse	88.5%	0	3643
<code>commons-attributes-api-2.2.jar</code> compared to <code>org.eclipse.jdt.ui.jar</code>	93.6%	0.6	28203
Previous test in reverse	97.6%	0	37660
<code>org.eclipse.jface.text.jar</code> compared to <code>org.eclipse.jdt.ui.jar</code>	88.5%	22.7	719561
Previous test in reverse	97.6%	3.0	820536

## 5. CONCLUSION

In this paper we have presented FiGD, an algorithm and implementation for detecting clones in compiled Java projects. Even when access to the source code is not available, FiGD is able to produce very accurate results in short periods of time by using a combination of previous approaches as well as custom optimizations. Source code for FiGD is released under the BSD license and is available by request. Included with the source code is full Javadoc documentation describing all methods and classes.

### 5.1 Review Goals and Contributions

Our main goals discussed in Section 1.3 are achieved with the design and implementation of our algorithm. We believe we are the first to approach the clone detection problem for software through a black box approach, giving the OSS community another tool for detecting IP violations.

### 5.2 Future Work

While we have shown that we are able to compare fingerprints quickly, there are still some possible optimizations that could be made in terms of generating each fingerprint. Making use of an advanced data structure (such as heaps) would provide us with faster searching than the current array-based implementation. This could theoretically reduce our worst-case running time for computing fingerprints to  $O(n \cdot \log n)$ . Together with stored pre-computation of all the known JAR files previously fingerprinted and stored offline, we believe that FiGD would operate significantly faster. We would also like to do more work on finding optimal  $n$ -gram sizes and how they impact the accuracy of the detector. Finally, we would like to expand the fingerprint to also include source code and plain text files as opposed to only considering Class files, and include these findings into a more advanced detection schema.

## 6. REFERENCES

- [1] Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue: CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code. *IEEE Trans. Software Eng.* 28(7): 654-670, 2002.
- [2] Clone Detection Literature - University of Alabama at Birmingham. <http://students.cis.uab.edu/tairasr/clones/literature/>. Accessed November 3, 2008.
- [3] Clone Digger. <http://sourceforge.net/projects/clonedigger/>. Accessed November 3, 2008.
- [4] Ira D. Baxter, Andrew Yahin, Leonardo Mendonça de Moura, Marcelo Sant'Anna, Lorraine Bier: Clone Detection Using Abstract Syntax Trees. *ICSM 1998*: 368-377. 1998.
- [5] JAR File Specification. <http://java.sun.com/j2se/1.5.0/docs/guide/jar/jar.html>. Accessed November 3, 2008.
- [6] Simian - Similarity Analyzer. <http://www.redhillconsulting.com.au/products/simian/index.html>. Accessed November 3, 2008.
- [7] Wikipedia - Bytecode. <http://en.wikipedia.org/wiki/Bytecode>. Accessed November 2, 2008.
- [8] Wikipedia - Class File. [http://en.wikipedia.org/wiki/Class\\_\(file\\_format\)](http://en.wikipedia.org/wiki/Class_(file_format)). Accessed November 3, 2008.
- [9] Arthur Choi. Java tip 49: How to extract java resources from jar and zip archives. <http://www.javaworld.com/javaworld/javatips/jw-javatip49.html>. Accessed October 30, 2008.
- [10] Black Duck Software. <http://www.blackducksoftware.com/protex>. Accessed October 30, 2008.
- [11] NetBeans.org. Classfile reader java documentation. <http://bits.netbeans.org/dev/javadoc/org-netbeans-modules-classfile/>. Accessed October 30, 2008.
- [12] Open Source Licenses - Free Software Foundation. <http://www.fsf.org/licensing/licenses/>. Accessed October 31, 2008.
- [13] Palamida Software. <http://www.palamida.com/products>. Accessed October 25, 2008.
- [14] HTTP Server Project. <http://httpd.apache.org/>. Accessed October 25, 2008.
- [15] Apache Tomcat. <http://tomcat.apache.org/>. Accessed October 25, 2008.
- [16] P. Bulychev, M. Minea, Duplicate code detection using anti-unification, in: Spring Young Researchers Colloquium on Software Engineering, SYRCoSE 2008, 2008, p. 4

# Integrating Privacy Requirements into Security Requirements Engineering

Saeed Abu-Nimeh  
Websense Security Labs  
San Diego, CA 92121  
sabu-nimeh@websense.com

Seiya Miyazaki  
Panasonic Corp.  
Tokyo, Japan  
miyazaki.seiya@jp.panasonic.com

Nancy R. Mead  
Carnegie Mellon University  
Pittsburgh, PA 15213  
nrm@sei.cmu.edu

**Abstract**—Security quality requirements engineering (SQUARE) is a structured methodology to address software security issues in early stages of the development lifecycle. It is possible to apply parts of this methodology to address privacy issues. This study presents the integration of privacy requirements into SQUARE. In addition, case studies are demonstrated to evaluate the efficiency of the modified model. Furthermore, alternatives to the existing security risk assessment techniques in SQUARE are suggested to make the model applicable to privacy.

## I. INTRODUCTION

Software developers often ignore security and privacy requirements or consider them to be a lower priority in the software production lifecycle. The highest priority during software production is typically given to functional requirements. However, nowadays with the ubiquity of online applications and other client-based software, protecting the privacy of users raises the need for systematic approaches designed solely for such purpose.

Security requirements engineering [1] aims to identify software security risks in early stages of the design process. Privacy requirements engineering [2] serves to identify privacy risks early in the design process. Recent research studies [3] have shown that privacy requirements engineering is less mature than security engineering and that underlying engineering principles give little attention to privacy requirements.

In addition, in [4] the authors claim that most of the privacy disclosures happen due to defects in the design, and are not the result of an intentional attack. Therefore, although security and privacy risks overlap, relying merely on protecting the security of users does not necessarily imply the protection of their privacy. For instance, health records can be secured from various types of intrusions; however, the security of such assets does not guarantee that the privacy of patients is secure. The security of such records does not protect against improper authorized access or disclosure of records.

### A. Security Quality Requirements Engineering

Security quality requirements engineering (SQUARE) is a structured methodology to address software security issues in the early stages of the development lifecycle. The technique consists of nine steps and generates categorized and prioritized security requirements [1].

- 1) Technical definitions are agreed upon by the requirements engineering team and project stakeholders.
- 2) Business and security goals are outlined.
- 3) In order to facilitate full understanding of the studied system, artifacts and documentation are created.
- 4) A security risk assessment is applied to determine the likelihood and impact of possible threats to the system.
- 5) The best method for eliciting security requirements is determined by the requirements engineering team and the stakeholders.
- 6) Security requirements are elicited.
- 7) Security requirements are categorized.
- 8) Security requirements are prioritized.
- 9) The security requirements are inspected to ensure consistency and accuracy.

The contribution of the present study is three-fold. First, we introduce the integration of privacy requirements into SQUARE. Then, we demonstrate case studies to evaluate the efficiency of our new model. Finally, we suggest improvements to the modified model by replacing the existing security risk assessments with privacy risk assessments, which we conjecture to be more suitable to address privacy issues.

The organization of the rest of the paper is as follows. In section II we discuss the related work. Section III describes the integration of privacy requirements into SQUARE. In section IV we demonstrate case studies to evaluate our new model. Section V illustrates the limitations of the risk assessment techniques in the current model and suggests alternatives that are more applicable to privacy. We conclude and motivate future work in section VI.

## II. RELATED WORK

As we mentioned earlier, privacy requirements engineering is less mature than security engineering [3]. In the following, we summarize studies that discuss security requirements as well as privacy requirements.

In [5], the authors introduced an approach hinging on the Common Criteria standard to handle security requirements at the early stages of software development. The approach relied on providing a security resources repository as well as integrating Common Criteria into the software lifecycle. The approach proposed a nine-step security requirements

engineering process that integrated well into other existing approaches.

In [6], the author developed a privacy risk management tool (PRMT) to investigate privacy risks in personal electronic health records (PEHR) and centralized electronic health records (CEHR). A qualitative privacy risk assessment approach was used in the study. The study showed how privacy risk assessment can be reduced to simple set of questions using qualitative methods.

In [7], the author summarized resources of privacy risk assessment approaches, namely web-based and association resources, that are related to the compliance with the Health Insurance Portability and Accountability Act (HIPAA). However, the list in the study is not comprehensive and is considered a starting point for healthcare organizations regarding conducting a privacy risk assessment.

In this study, we modify a security requirements engineering model (SQUARE) and introduce steps in the model that enables it to elicit privacy requirements. In addition, we suggest alternatives to the existing security risk assessment techniques in SQUARE to make it applicable to privacy. The following sections discuss the process in more detail.

### III. PRIVACY REQUIREMENTS INTEGRATION INTO SQUARE

We propose a privacy requirements elicitation technique (PRET) [8], which is based on SQUARE. The tool helps software engineers and stakeholders elicit privacy requirements using a computer-aided approach. PRET uses a questionnaire to elicit information that the requirements engineers and stakeholders complete. The tool contains a database of privacy requirements that is searched to utilize the input from the questionnaire and provides results. Figure 1 depicts the integration of PRET into SQUARE. The first four steps in SQUARE remain the same. In the fifth step, when elicitation techniques are chosen, PRET can be selected to elicit privacy requirements. Here the PRET process starts; the questionnaire is answered and the privacy requirements are elicited. Finally, the privacy requirements are verified and fed back to SQUARE. In SQUARE, the privacy requirements are categorized, prioritized, and inspected in the seventh, the eighth, and the ninth steps respectively. In the following subsections, we describe the questionnaire design, discuss the various sources used to identify privacy requirements, and illustrate the process involved in requirement elicitation.

#### A. Questionnaire Design

Privacy Seal Programs [9] and the OECD Privacy Statement Generator [10] are used to prepare the questionnaire. The OECD generator is a tool that provides users with useful input in the development of a privacy policy and statement. Using the generator and other privacy seal policies, such as TRUSTe and PrivacyMark, 10 questions are included in the questionnaire as shown in Table I.



Fig. 1. Integration of PRET into SQUARE

#### B. Identification of Privacy Requirements

Privacy requirements are collected from multiple sources, which are generally various publicly available privacy laws and principles. In addition, we apply misuse cases to identify privacy requirements. The following outlines each of these approaches:

1) *Privacy laws and principles*: To identify privacy requirements, six privacy principles and laws are studied, from which a subset of privacy requirements are selected. Due to space constraints, the interested reader can refer to [8] and the references therein for a detailed overview. The laws and principles are as follows:

- OECD Guidelines on the Protection of Privacy
- The European Commission’s Directive on Data Protection
- Japan’s Personal Information Protection Act
- Privacy laws in the US
  - Privacy Protection Act
  - Video Privacy Protection Act
  - CA–SB–1386 (California)
  - Family Educational Rights and Privacy Act
  - Health Insurance Portability and Accountability Act
  - Children’s Online Privacy Protection Act
- Common Criteria
- W3C Web Services Architecture Requirements

2) *Misuse cases*: Misuse cases are used to elicit requirements. The idea behind them is to document and decide how software should act proactively to malicious activities. First, a normal use case is assumed. Then, malicious parties and activities are added to the use case. Afterwards, the relationships among the use cases and the misuse cases are linked. This whole process has proven to be useful in mitigating future attacks.

#### C. Decision Process

A decision tree of requirements is built to traverse multiple combinations of question paths. The introduction of



TABLE I  
QUESTIONS INCLUDED IN QUESTIONNAIRE

Question	Response
1. Does the service provider process personal information?	Yes / No
2. In which country or area is the service provided?	USA / EU / Canada / Japan / Other
3. What type of service provider? 3.1. If Industrial, does the service provider belong to any of these fields? 3.2. If Governmental, does the service provider belong to any of these fields? 3.3. Is the purpose of the service related to journalism, literary work, academic studies, religious activities, or political activities?	Industrial / Governmental / Academic / Other Medicine / Communication / Education  Military branch / Non-military branch / Research Body  Yes / No
4. What kind of personal information does the service provider process?	Point of Contact / Social Identification / Personal Identity Data / Demographic Information / Age, Education / Health Information / Financial Information / Personal Information of Children / Other Sensitive Personal Data
5. How does the service provider obtain personal information?	Provided by users / Provided by third parties / Collected automatically from users / Collected automatically from third parties
6. Where does the service provider store personal information?	Client Side / Server Side / The Third Party Client Side / The Third Party Server Side
7. How long does the service provider store personal information?	Does Not Store / One Transaction / Certain Period of Time / Forever
8. Does the service provider use personal information for another purpose?	Yes / No
9. Does the service provider share personal information with others?	Yes / No
10. What privacy protection level does the service provider set?	High / Mid / Low

subsequent questions is based on the answers to the current question. While the user goes through different nodes in the decision tree, a different set of questions is introduced. Several constraints are checked to ensure that privacy requirements dedicated to certain areas, (e.g., in the US or the EU), are met. In addition, each one of the requirements is assigned a priority based on its source. For instance, requirements derived from laws have higher priority than requirements derived from principles and misuse cases.

#### IV. EVALUATION CASE STUDIES

We evaluate our model using two pseudo-software development projects; an auto insurance service and a health care ring. In Table II, we show the answers to the questionnaire for the auto insurance service. Then, we show the corresponding privacy requirements elicited by PRET in Table III. Similarly, the health care ring's answers to the questionnaire are shown in Table IV and the elicited privacy requirements are shown in Table V.

We evaluate the tool's results based on its requirement achievement against several functional and non-functional requirements. Further, we compare the tool with nine other elicitation techniques using various measures reported in [11]. Moreover, we seek privacy experts' opinions on the quality of the results. PRET outperforms all rivals with a few minor changes requested by experts (see [8]).

#### V. SECURITY RISK ASSESSMENT LIMITATIONS

The same security risk assessment techniques used in SQUARE are applied to privacy in PRET (see Figure 1). We

TABLE II  
AUTO INSURANCE SERVICE QUESTIONNAIRE

Question	Answer
1	Yes
2	USA
3	Industrial
3.1	-
3.2	-
3.3	No
4	Point, Social, Demographic, Age
5	Provided by users, Provided by third parties
6	Server side
7	Forever
8	No
9	No
10	Mid

TABLE IV  
HEALTH CARE RING QUESTIONNAIRE

Question	Answer
1	Yes
2	Japan
3	Industrial
3.1	Medicine
3.2	-
3.3	No
4	Point, Demographic, Age, Health
5	Provided by users
6	Server side
7	Certain Period of Time
8	No
9	Yes
10	High

believe that this is a limitation in PRET, as security risk assess-

TABLE III  
AUTO INSURANCE SERVICE RESULTS

Privacy Requirements	Derivation	Explanation	Priority Level
The service architecture shall describe privacy policy statements and enable a user to access them.	W3C-AR020.1,20.3	Personal data usage(Q1, Q2)	Mid
Before collecting personal data, the data controller shall specify the purpose.	OECD-PP-P9	Personal data usage(Q1)	Mid
The service provider shall limit the collection of personal data and obtain such data by lawful and fair means.	OECD-PP-P7	Personal data collection (Q6)	Mid
The system network communications must be protected from unauthorized information gathering and/or eavesdropping.	Misuse-case-1	Personal data collection (Q6)	Mid
The system should have functional audit logs and usage reports without disclosing identity information.	Misuse-case-2	Personal data collection (Q6)	Mid
The system shall have strong authentication measures in place at all system gateways and entrance points.	Misuse-case-3	Personal data storage (Q7)	Mid
Personal data should be protected by reasonable security safeguards against such risks as loss, unauthorized access, destruction, use, modification or disclosure of data.	OECD-PP-P11	Personal data storage (Q7)	Mid
Personal data shall be accurate, complete and kept up-to-date, if it is possible.	OECD-PP-P8	Personal data storage (Q7)	Mid
The system shall provide a mechanism by which users can verify their data.	OECD-PP-P13	Personal data storage (Q7)	Mid
The system shall provide a data backup mechanism.	Misuse-case-4	Personal data storage (Q7)	Mid
The system shall have a verification process to check whether there is a disclosure agreement between the third party and the person.	Misuse-case-5	Personal data collection from the third party (Q5)	Mid
The service provider shall report to all the customers if the privacy information is breached.	CA-SB-1386	Breach report in JP, USA (Q1,Q2, Q3)	High

ment cannot substitute for privacy risk assessment. Although security risk assessment and privacy risk assessment overlap, they are different. Security protects systems' resources, including software, storage, networks, and users. However, privacy concentrates on data protection, which includes the application of various policies and procedures to collect and protect data.

The goals of a security risk assessment include the implementation of authentication and authorization systems, which can be done by building firewalls, enforcing levels of authority, and generating audit trails and logs. In addition, security risk assessments ensure the protection of network security, physical security, and system security.

However, the goals of a privacy risk assessment relate to policies and procedures. The focus is on the nature of data collected, the purpose of data collection, and the procedures for obtaining an individual's consent. Further, the privacy risk assessment takes into account the necessity and accuracy of data, and compliance to regulations. Also the assessment ensures that standards exist for development projects and auditing compliance. Furthermore, the assessment checks authorization and authentication requirements, risks of theft, modification, or disclosure and mitigation procedures, third party vulnerabilities, and disclosure incident procedures [12].

SQUARE relies on two risk assessment techniques in step 4, namely the Risk Management Guide for Information Technology Systems (NIST SP 800-30) [13] and Yacov Haimes's Risk Filtering, Ranking, and Management Framework (RFRM) [14]. The RFRM approach contains eight phases, some of which were found to be out of scope. As such, only two

relevant two phases of RFRM are included in SQUARE: phase III, Bicriteria filtering and ranking, and phase IV, multicriteria filtering and ranking.

NIST's model for risk assessment is broken into nine steps, each with an output that serves as the input to the next step. SQUARE excludes steps 1, 8, and 9 in NIST, as they are irrelevant or redundant. Therefore, the steps included in SQUARE are: threat identification, vulnerability identification, control analysis, likelihood determination, impact analysis, and risk determination. Apparently, the risk assessment in SQUARE corresponds to the system under analysis. Most importantly, the risk assessment should categorize the likelihood and impact of the major threats to the system [1].

Based on the above discussion between security and privacy risk assessment, we conjecture that the quality of the privacy requirements elicited by PRET will improve if the existing security risk assessment techniques are replaced with, or combined with, privacy risk assessment techniques. Consequently, we suggest the introduction of privacy impact assessment (PIA) in PRET.

According to [15], "PIA is a comprehensive process for determining the privacy, confidentiality and security risks associated with the collection, use and disclosure of personal information. It also defines the measures used to mitigate and, wherever possible, eliminate the identified risks. The PIA process ensures that measures intended to protect privacy and ensure the confidentiality and security of personal information are considered at the outset of any new program or service delivery initiative. A PIA also communicates to the public

TABLE V  
HEALTH CARE RING RESULTS

Privacy Requirements	Derivation	Explanation	Priority Level
The service provider shall describe privacy policy statements and enable a user to access them.	W3C-AR020.1,20.3	Personal data usage(Q1, Q2)	Mid
Before collecting personal data, the service provider shall specify the purpose.	OECD-PP-P9	Personal data usage(Q1)	Mid
The service provider shall obtain prior consent of the person, except for following cases; (1) handling of personal information is based on laws; (2) handling of personal information is based on necessity for the protection of the life, body, or property of an individual; (3) handling of personal information is based on necessity for improving public hygiene or promoting the growth of children; (4) handling of personal information is based on necessity for cooperating with a state institution, a local public body, or an individual or entity entrusted by one in executing the operations prescribed by laws.	PIPA-Article-16	Personal data usage in JP (Q1, Q2, Q3)	High
The service provider shall handle personal information within the scope for the purpose of usage.	PIPA-Article-16	Personal data usage in JP (Q1, Q2, Q3)	High
The service provider shall limit the collection of personal data and obtain such data by lawful and fair means.	OECD-PP-P7	Personal data collection (Q6)	Mid
The system network communications must be protected from unauthorized information gathering and/or eavesdropping.	Misuse-case-1	Personal data collection (Q6)	Mid
The system should have functional audit logs and usage reports without disclosing identity information.	Misuse-case-2	Personal data collection (Q6)	Mid
The system shall have strong authentication measures in place at all system gateways and entrance points.	Misuse-case-3	Personal data storage (Q7)	Mid
Personal data shall be accurate, complete and kept up-to-date, if it is possible.	OECD-PP-P8	Personal data storage (Q7)	Mid
The system shall provide a mechanism by which users can verify their data.	OECD-PP-P13	Personal data storage (Q7)	Mid
The system shall provide a data backup mechanism.	Misuse-case-4	Personal data storage (Q7)	Mid
The service provider must take necessary and proper measures for the prevention of leakage, loss, or damage, and for other control of security of personal data.	PIPA-Article-20	Personal data storage in JP (Q2, Q7)	High
The service provider shall disclose personal data only with the consent of data subject or by the authority of law.	OECD-PP-P10	Personal data sharing (Q9)	Mid
The service provider shall enable delegation and propagation of privacy policy to the third parties.	W3C-AR020.5	Personal data sharing (Q9)	Mid
The service provider shall gain consensus from users what data they are sharing.	PIPA-Article-23	Personal data sharing in JP (Q2, Q3, Q9)	High
The service provider shall report to all the customers if the privacy information is breached.	CA-SB-1386	Breach report in JP, USA (Q1,Q2, Q3)	High
The system should provide anonymity. Anonymity means other users or subjects are unable to determine the identity of a user bound to a subject or operation.	CC-FPR-ANO	Privacy enhancing technology usage (Q10)	Low
The system should provide pseudonymity. Pseudonymity means a set of users and/or subjects are unable to determine the identity of a user bound to a subject or operation, but that this user is still accountable for its actions.	CC-FPR-PSE	Privacy enhancing technology usage (Q10)	Low
The system should provide unlinkability. Unlinkability means users and/or subjects are unable to determine whether the same user caused certain specific operations.	CC-FPR-UNL	Privacy enhancing technology usage (Q10)	Low
The system should provide unobservability. Unobservability means users and/or subjects cannot determine whether an operation is being performed.	CC-FPR-UNO	Privacy enhancing technology usage (Q10)	Low
The system should provide unobservability, which requires that users and/or subjects cannot determine whether an operation is being performed.	CC-FPR-UNO	Privacy enhancing technology usage (Q10)	Low

how their privacy is protected and their information kept confidential and secure from unauthorized access.”

A very well known PIA tool is the one used by the US Internal Revenue Service (IRS) [16]. Comparing the NIST security risk assessment procedure with the IRS PIA clearly shows that the latter is more applicable to privacy. The following lists summarize a few of the procedures in both approaches.

- 1) NIST security risk assessment
  - Threat identification
  - Vulnerability identification
  - Control analysis
  - Likelihood determination
  - Impact analysis
  - Risk determination
- 2) IRS privacy impact assessment
  - Data description
  - Data sources
  - Data collection process, data accuracy, data completeness, and data currentness
  - Data comprehensiveness and documentation
  - Data access description, access procedures, access controls, and access responsibilities
  - Access levels and restrictions
  - Authorized access misuse
  - Shared data restrictions and controls
  - Data relevancy and necessity
  - Possibility of data derivation and aggregation
  - Protection and control of consolidated data
  - Data retrieval
  - Equitable treatment of users
  - Data retention and disposal
  - User monitoring and protection against unauthorized monitoring

Previous research [17] proved that PIA works well in combination with other risk assessment techniques. PIA helped to identify the data sensitivities of vote verification systems, while other risk assessments were used to identify the full spectrum of threats to these systems. This demonstrates that both security and privacy risk assessments assess risk from different perspectives, hence one cannot substitute for another.

## VI. CONCLUSIONS AND FUTURE WORK

The present study discussed the integration of privacy requirements into an existing security requirements engineering model SQUARE. SQUARE is a structured methodology to address software security issues in early stages of the development lifecycle. The study proposed a tool, namely PRET, to address privacy issues in the development lifecycle.

PRET relies in some of the steps on SQUARE, but introduces extra steps to be applicable to privacy issues. We evaluated our model using the pseudo-software projects of an auto insurance service and a health care ring. In addition, we compared the tool with nine other elicitation techniques. PRET

outperformed all rivals with a few minor changes requested by consulted privacy experts.

Furthermore, based on previous research, we showed that existing security risk assessment techniques are not suitable for privacy. Therefore, we suggested replacing, or combining, current risk assessment techniques in PRET with a privacy impact assessment model, such as the IRS PIA. We conjecture that the quality of privacy requirements will improve with these changes. We intend to report case studies in subsequent publications.

The future work will explore the integration of the IRS privacy impact assessment into SQUARE. We will compare the quality of the results by replacing the current risk assessment techniques with the IRS PIA and then combining them with IRS PIA. Further, case studies will be explored to gauge efficacy of the modified model.

## REFERENCES

- [1] N. R. Mead, E. Hough, and T. Stehney, “Security quality requirements engineering (SQUARE) methodology,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2005-TR-009, 2005, <http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.html>.
- [2] A. Chiasera, F. Casati, F. Daniel, and Y. Velegrakis, “Engineering privacy requirements in business intelligence applications,” in *SDM '08: Proceedings of the 5th VLDB workshop on Secure Data Management*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 219–228.
- [3] S. L. Pfleeger and C. P. Pfleeger, “Harmonizing privacy with security principles and practices,” *IBM Journal for research and development*, accepted.
- [4] A. Adams and M. A. Sasse, “Privacy in multimedia communications: Protecting users, not just data,” in *Proceedings of IMH HCI'01*. Springer, 2001, pp. 49–64.
- [5] D. Mellado, E. Fernández-Medina, and M. Piattini, “A common criteria based security requirements engineering process for the development of secure information systems,” *Comput. Stand. Interfaces*, vol. 29, no. 2, pp. 244–253, 2007.
- [6] M. Madsen, “EHR privacy risk assessment using qualitative methods,” in *HIC 2008 Conference: Australia's Health Informatics Conference*, 2008.
- [7] J. C. Dennis, “Leading the HIPAA privacy risk assessment,” in *AHIMA Convention Proceedings*, 2001.
- [8] S. Miyazaki, N. Mead, and J. Zhan, “Computer-aided privacy requirements elicitation technique,” *Asia-Pacific Conference on Services Computing*, vol. 0, pp. 367–372, 2008.
- [9] B. K. Markert, “Comparison of three online privacy seal programs,” SANS Institute, Tech. Rep., 2002.
- [10] OECD and Microsoft Corp., “OECD privacy statement generator,” 2000. [Online]. Available: <http://www2.oecd.org/pwv3/>
- [11] J. H. Allen, S. Barnum, R. J. Ellison, G. McGraw, and N. R. Mead, *Software Security Engineering: A Guide for Project Managers*. Addison-Wesley, 2008.
- [12] T. Mitrano, D. R. Kirby, and L. Maltz, “What does privacy have to do with it? privacy risk assessment,” in *Security Professionals Conference*, 2005, presentation.
- [13] National Institute of Standards and Technology, “Risk management guide for information technology systems,” 2002. [Online]. Available: <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>
- [14] Y. Y. Haimes, *Risk Modeling, Assessment, and Management*, 2nd ed. Wiley-Interscience, 2004.
- [15] Statistics Canada, “Privacy impact assessment,” 2008. [Online]. Available: <http://www.statcan.gc.ca/about-apercu/pia-efrvp/gloss-eng.htm>
- [16] Internal Revenue Service, “Model information technology privacy impact assessment,” 1996. [Online]. Available: [http://www.cio.gov/Documents/pia\\_for\\_it\\_irs\\_model.pdf](http://www.cio.gov/Documents/pia_for_it_irs_model.pdf)
- [17] R. R. Heckle and S. H. Holden, “Analytical tools for privacy risks: Assessing efficacy on vote verification technologies,” in *Symposium On Usable Privacy and Security*, 2006, poster.

# iPass: An Integrated Framework for Educating, Monitoring and Enforcing Password Policies for Online Services

Dhananjay Kulkarni\*, Diana Ciric<sup>†</sup> and Fernanda Zulkarnain\*

\*Boston University, Department of Computer Science - Metropolitan College  
808 Commonwealth Avenue, Boston, MA, USA

<sup>†</sup>Faculty of Organization Sciences - Department for Information Systems and Technologies  
Jove Ilica 154, Belgrade, Serbia

**Abstract**—Creating an account (with username and password) is the first line of defense for any online service. Though the process for specifying passwords has been around, not much has been done in implementing policies so that users can satisfactorily create a strong, yet easy-to-remember password. Users prefer simple passwords, but they are at a risk of being guessed by hackers. On the contrary, companies may prefer a strong password policy so that their resources are protected from unauthorized access. Usability and security become conflicting goals while implementing password policies, since there is a risk of losing customers if the process is too cumbersome. We propose a novel framework, called *iPass*, that takes multiple objectives into account, and is tunable to customer satisfaction. Our goal is to guide users in creating *secure* passwords, and remind them to update passwords based on password strength and usability. Our approach emphasizes on ‘education’, ‘monitoring’, and ‘usability-aware’ enforcement of passwords. We also provide appropriate feedback, and acknowledge user efforts in creating passwords. The prototype we have developed is simple, practical, and effective in addressing the problem.

## I. INTRODUCTION

As part of conducting their business, online services such as banks [1], auction sites [3], stock trading portals [12], and news agencies [9] require users to create a username and password before using their services. Usually the username and password is the first line of defense against unauthorized access to the company’s ‘resources’, while providing the flexibility of allowing the user to access their accounts online. However, this flexibility of accessing information and using services online comes with a cost. The cost for the user is the time that he/she has to spend in creating and remembering the username and the password. The cost for the online service provider includes implementing the password policy, and educating the users about password usage.

Most services also require users to provide additional personal information (birthdate, sex, address, etc.) or service-specific information (bank account details, shipping address, SSN). Login and password, thus not only protect hackers from obtaining personal or financial information, but also information that a legitimate user would access and transactions that could be made using that account. This means that users should be using a password that is hard to guess, and those that are resistant against various types of attacks.

A *strong* password is one that is very difficult to guess, or hack. A *simple* password is one that is easy to remember, but has a possibility of being too easy to guess. In general, passwords should be such that they are easy to remember, but also consistent with the password policy defined by the organization. A *secure* password is one that is both, *simple* and *strong*.

Although using usernames and password has been around since the inception of online services, not much has been done in providing a practical framework that can address education, usability and implementation of password policies. As we will see in the next section, this is a very ambitious goal because it requires balancing customer satisfaction (or usability) of the systems and security of the system – which are often conflicting goals.

### A. Motivation

Majority of Internet users have to manage on average about 5-7 passwords [11]. Since remembering all passwords is challenging, most users prefer to create passwords that are *simple*. Also, creating a simple password is less time consuming for the user. When accessing the service, the user can quickly type the username and password, rather than refer to a password written on paper (which has its own security risks). Hence, at one spectrum, users have a valid reason to choose a simple password.

Online service providers would like to enforce a *strong* password policy, because systems providing these services are vulnerable to various types of attacks [15]. It is obvious that unauthorized access to its resources poses a big security risk related to confidentiality and integrity of information. For example, if someone hacks into a customer account and transfers a larger amount of money to his off-shore account, the bank may be liable for not doing enough to protect its customers. There is also the risk of a hacker using a compromised account (for example a root account) to obtain, or disable other valid user accounts. To mitigate such risks, the service provider has valid incentive to impose a *strong* password policy. For example, eBay [3] provides a hyper-link, which directs the user to a help page creating a password. Google [5] provides a ‘password meter’ that shows the strength of a password string.

Since the above two motivating examples lead to conflicting goals, there are risks involved in forcing, or not forcing a password policy. So, what are the risks involved if the service provider forced all users to use create a *strong* password, which may be time-consuming to create, and hard to remember? We argue that customer satisfaction to using the online service is at a risk, since most users prefer using a service that is user-friendly. There is also a risk of losing customers to competitors, for example, online bidding sites eBay [3] and uBid [13]. Customers also may feel annoyed or challenged, if several of their password choices are rejected.

This brings us to the problem of risks involved in *not* forcing a password policy. Allowing users to create simple passwords for example, is vulnerable to a dictionary attack [2], or just guessing. Although the usability (and complexity of remembering a password) will be improved, the security risks involved outweigh the advantages.

### B. Challenges

With more than 1.5 billion Internet users [6] it is a daunting task to address the problem of password policies. We see this as a 2-wall problem [14]. First, the user does not know *what* is a *strong* password and the risks involved in not using one. Second, the user does not know *how* to create a *strong* password. We will discuss the challenges below.

**Education:** Users are unaware of the risks, and use of simple passwords. We argue that the education should be consistent with the password policy of the company, and allow a user to fully understand the risks and conditions that can make the account insecure. Second, the education should be not as complicated to understand. For example, the use of a password meter [5] is useful, but the user does not know why the meter says that the password is weak or strong. Examples, and user-assisted learning is a better approach to educate the user on why a change in the password string may strengthen or weaken the password.

**Monitoring:** Although education is the first step, the users may still create passwords that will be easy to guess or prone to some kind of dictionary attack. It should be the responsibility of the online service provider to monitor passwords, including any modifications in the future. The challenge lies in implementing the tools, and making sure that the tools can identify potential vulnerabilities in user passwords. With appropriate feedback, such vulnerabilities can be eliminated by the user by modifying the password.

**Balancing customer satisfaction and security goals:** Usability of the system (in terms of the customer satisfaction) and the security policy of a company are conflicting goals. The challenge is to create a balanced environment, where a user will learn, and in this process the user will be willing to take measures to secure the password. Mere forcing a policy has disadvantages as we have pointed out earlier. We believe that the security goal can be achieved by combining education, monitoring and providing appropriate feedback to the user to improve the usability of the system.

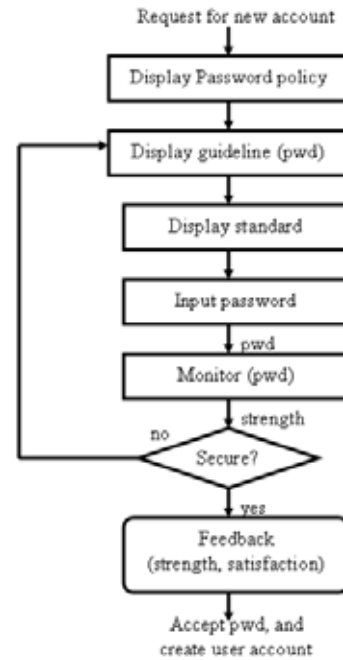


Fig. 1. The iPass Framework

## II. OUR APPROACH

Figure 1 shows the conceptual model of our iPass framework. Many Internet users are unaware of the meaning or need for *strong* passwords, so the first step toward successful implementation of password policy is education. Through education, and display of the policy and relevant guidelines conveniently to the user, we try to make users aware of how passwords can be secured. Our goal would be to guide a user to creating a *secure* password - one that is strong and easy to remember. Monitoring the password and reporting potential vulnerabilities is also important, since hackers are known to stay one step ahead in guessing passwords. Our goal in monitoring is two fold, making sure that the passwords comply with the policy, and giving a feedback to the user based on the current strength (and vulnerabilities) of the password. Since this framework addresses multiple conflicting goals (usability and the security of passwords), we extend the approach to make it tunable to user satisfaction level. Thus, our framework can control user satisfaction and help maintain system security simultaneously. We describe the details in following sections.

### A. Educating Users about Password Policy

As shown in the iPass framework, the display of policy, guideline and standard all are the educational part of our approach. The password policy and the standards remain static and can be developed off-line. In many organizations, the policy development is done in various stages. It is important to display the main parts of the policy (the objective, audience, approvals, and definitions) on the registration page itself. This would help not only insiders, but also outside users to know that the company is serious about the password policy.

The guidelines are the most important part of helping a user to create a *secure* password. In our approach, we first display a default guideline, which may include examples of some bad, simple, and strong passwords. It will also include 2-3 examples of *secure* passwords that are consistent with the password policy. At least 1 such password is just consistent with the standard, which means that it is sufficiently *strong* and also meets the standard shown to the user.

It is also worth noting that the guidelines are not static, they are updated as a function of the user input. We dynamically update the guidelines to ‘guide’ the user towards creating a secure password. For example, if the user tried to create an initial password (say ‘t2ylayhw’) which has all the requisite characters, but misses a digit or a special character, then we update the guidelines page. In this updated guidelines, we display possible choices for a *secure* password which are derived from the user-input *pwd* (for example, ‘t2ylayhw19@’, ‘t2ylayhw@’). We believe that this technique, recognizes the user’s efforts in creating passwords, contrary to the approach where a password maybe just rejected since it did not meet the standards. By observing our updated guidelines, the user can easily pick one of the secure passwords, or construct one in his or her mind that will be *simple, yet secure*.

### B. Monitoring Password Strength

Since our goal is to assure that all user passwords are *secure*, monitoring them and validating that they conform to the password standard becomes a necessity. The goals of a password validator used to monitor user-created passwords are many. First, it must check if the given password has the minimum number of characters, digits, and special characters. Second, the validator should check if any substring is a dictionary (or colloquial word), since all such passwords are vulnerable to dictionary attacks. Third, the validator may also check if the password includes other information that is provided by the user, for example last name, social security number, or place of birth. Such information, though not available in a dictionary, may very well be obtained by social-engineering techniques and later used to hack passwords. Only when the given password passes all these tests, *pwd* can be labeled as *secure*.

The secondary goal of the monitoring process is to determine the *strength* of a password. Any scheme, such as the one developed at Microsoft [10] or one used by Google Accounts [4] may be used. The objective is to rate the user password on a scale, say 1-10 that indicates how difficult it is to guess the password. In our work, 10 indicates the maximum strength. As we explain in the next section, this rating or strength is used to assist in two ways. First, if the password strength is below a certain threshold (say  $\tau$ ) then the password is not secure. In this case we allow the user to update the password string as explained above, and guide the user towards a more secure password. Secondly, this rating can be used for the continued security of the passwords, even after they are successfully created. For example, passwords that are *secure* but not rated high enough can be considered potential

passwords that should be updated by users in the recent future. Stronger passwords (for example those rated as 10) may be allowed to be retained for a longer duration of time. Our work is the first in providing such flexibility in updating passwords.

### C. Usability-aware Enforcement

As discussed earlier, security of a system and usability (or customer satisfaction) are conflicting goals. Most services that a relatively secure have additional levels of security, just to make sure that only legitimate users are accessing their resources. For example, Bank of America [1] employs ‘passcode’, a security ‘image’, and a ‘security question’ for user authentication. Discussion of their scheme is beyond the scope of our work, but it is easy to see that a user needs to spend additional time every time he needs to use the service. We use such user-interaction time and experience as an abstract measure of customer-satisfaction level. Other metrics, such as effectiveness of user-computer interaction may also be used.

After establishing the strength of a secure password, we wish to estimate when is the next time that the user should update the password. This estimation is important for two reasons. Firstly, we want to keep the passwords secure, and update them as often, so they can remain resistant against guessing or brute-force attacks. Secondly, we want to do not wish to request the users to update passwords too often. Requesting passwords to be updated, especially using an elaborate framework is cumbersome and will only result in lowering the customer-satisfaction. For this reason, we let the user participate in how often they should be reminded. In current work, we just estimate that the user be reminded of password change after  $K \times (\sigma/C)$  days, where  $K$  is a constant,  $\sigma$  is the strength of the password, and  $C$  is the user-specified customer-satisfaction. We allow the user to specify  $C$  at the end of creating his or her new account. Our implementation section makes this process more clear.

For example, assuming  $K = 100$ , a user who created a new password with  $\sigma = 5$  and  $C = 10$ , will be reminded after 50 days that the password needs to be updated. Another user, who created a password with  $\sigma = 5$  and  $C = 5$  would be reminded after 100 days.

This technique is in lieu with our goals, since we do not wish to annoy users (especially users that have shown low customer-satisfaction) by requesting password updates often. Remember that online service providers are at a risk of losing the customers if they are dissatisfied, and more often than not, the competitors are happy to accept new users. Our technique is simple, yet effective in balancing the security and usability of the system.

We also allow the user to override the above estimate, and specify the number of days that he or she should be reminded for updating passwords. For example, some users may know that their password is secure enough, yet, they would like to update it often. In this case the user would ignore the estimate, and just specify the number of days after which a reminder will be set out. This makes the job of the framework even easier, since the user himself has determined the current and future usability of the system.

### III. DESIGN AND IMPLEMENTATION

We have designed and implemented a prototype implementation of our iPass framework. Currently, this service is hosted at [7], and we encourage you to browse the service. You will notice that the 5 parts of the framework: (a) the policy, (b) guidelines, (c) standard, (d) the explanation of a strong password, and (e) the registration form are all on the same page. As stated earlier about iPass framework, we wish to educate the user *while* the password is being created. This simple approach of displaying, and providing appropriate feedback is effective. The question: ‘Why is it good to have a strong password?’, is just to make the user aware of the meaning, and the risk involved if a strong password is not used. Though our current implementation is very basic, we foresee that one can include other details here, but not limited to the following: a link to statistics about how many people use strong or weak passwords, what is the average rating of the passwords used, or an explanation of dictionary attack.

We used the following standard:

- Password must be between 8-15 characters
- Password must have 1 or more symbols: @, #, \$, %, &
- Password cannot have 3 consecutive number such as ‘123bu’, ‘angel678’, etc.
- Password cannot contain white spaces

Our sample policy, guidelines and standards are also discussed on the webpage [7]. Apart from the guidelines, we also dynamically generate a set of possible secure passwords (only if the first password entered by the user does not qualify as being secure). For example, if the first password entered is ‘qijwaethe’ then we display the guideline: ‘You can make your password secure by adding a digit or a special symbol, example ‘qijwaethe@’ will make it secure’. Passwords that do conform to a given standard are also given appropriate feedback. For example, if the user provided ‘abc123hij456’ then we display the guideline: ‘Consecutive characters or digits are not allowed. Please consider revising the password string. For example, ‘a1b2c3h4i5j6’ would be a secure password’.

Currently, we have developed a validator that checks that the given password meets the standard, or not. The validator is also able to perform sanity checks to ensure that the password does not contain personal information, such as last name, first name, social-security number or the credit-card number. Our validator is extendable, and we plan to add more rules to check for if passwords contain commonly obtainable information, such as spouse’s name or city name. Another possibility is to use our framework along with LC5 [8], which is a popular password auditing and recovery application.

Once we accept the password, we take the user to a page that lists the password strength, ranging between 1-10. We also ask the user the question: ‘What is the satisfaction level during the account registration phase?’ The user is allowed to enter a number between 1-10, where a higher value indicates better satisfaction. Once the user clicks the submit button, we estimate the number of days after which we would like to remind the user about changing the password. The user may over-ride this estimate by directly specifying the number of days. If the user specifies, then we assume that the user

indicates his choice based on the experience he has had during the registration process. Customer with good satisfaction-level, for example, would not mind getting reminded after 6 months. Customers who do not wish to go through this process more often, would opt for a longer (for example, 365 days) as a reminder option. We plan on using some survey statistics, and techniques in human-computer interaction to derive a more accurate estimate.

As stated above, a password is accepted only if it meets the standards. This we believe is a very important requirement for any online service, and hence iPass enforces the policy by default. Our framework does more than this, because we continually monitor and remind users to update passwords, not when they forget the password, but based on iPass estimation. Since we include user-satisfaction as well as password strength in sending the reminders, we also reduce the possibility of brute-force attacks. As passwords are changed more frequently, it becomes harder for the hacker to guess a target string. Hence, iPass does balance the security requirements as well as the customer-satisfaction by what we call *usability-aware enforcement* – we do not enforce too strictly if we feel that the usability might be affected. Our study shows that this is indeed a better model than forcing users. We are conducting another survey in this direction, and the results will be posted online, which the new users can also consult. For example, average customer-satisfaction of users who have chosen a *secure* password.

### IV. CONCLUSION

It is important to enforce policies that will make it mandatory to use strong passwords, and hence mitigate the risks of dictionary or brute-force attacks. However, the process needs to provide satisfactory customer satisfaction (or usability), else there is a risk of losing valuable customers. We have developed a novel framework, called *iPass*, which can balance usability and security goals. Our approach emphasizes that ‘education’, ‘monitoring’ and selectively ‘reminding’ the users is an effective way to enforce password policy. By encouraging user participation, and acknowledging their efforts in the password creating process, the users would be more consistent in creating and maintaining secure passwords.

### REFERENCES

- [1] Bank of america, <http://www.bankofamerica.com>.
- [2] Dictionary attack (wikipedia), [http://en.wikipedia.org/wiki/dictionary\\_attack](http://en.wikipedia.org/wiki/dictionary_attack).
- [3] ebay, <http://www.ebay.com>.
- [4] Google accounts, <https://www.google.com/accounts/newaccount>.
- [5] Google, <http://www.google.com>.
- [6] Internet world stats, homepage, <http://www.internetworldstats.com/stats.htm>.
- [7] ipass project, <http://www.geocities.com/ipass.project/>.
- [8] L0phtcrack (wikipedia), <http://en.wikipedia.org/wiki/l0phtcrack>.
- [9] The new york times, <http://www.nytimes.com/>.
- [10] Password checker, <http://www.microsoft.com/protect/yourself/password>.
- [11] Password research, <http://passwordresearch.com/stats/statistic246.html>.
- [12] Td ameritrade, <http://www.tdameritrade.com>.
- [13] ubid, <http://www.ubid.com>.
- [14] S. Greene. *Security Policies and Procedures: Principles and Practices (Prentice Hall Security Series)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2005.
- [15] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.



# Improving Natural Language Specifications with Ontologies

Sven J. Körner

Institute for Programming and Data Structures  
University of Karlsruhe  
76128 Karlsruhe, Germany  
Email: koerner@ipd.uka.de

Torben Brumm

Institute for Programming and Data Structures  
University of Karlsruhe  
76128 Karlsruhe, Germany  
Email: brumm@ipd.uka.de

**Abstract**—User requirements are usually written in textual form. Dealing with natural language textual specifications is complex. Analysts<sup>1</sup> have to cover all aspects of the requirements engineering process when working with the customer, such as work flows, psychological issues, and linguistic problems. We show how analysts can be supported during requirements elicitation and documentation. We present an approach to improve natural language requirements specifications using ontologies. We demonstrate by several examples from real requirements how an ontological reasoner called RESI<sup>2</sup> can uncover gaps, inaccuracies, ambiguities and ask the user to clarify them. In many cases, it supports the analyst by giving a small number of reasonable suggestions to choose from. The implementation of RESI is currently in progress.

## I. INTRODUCTION

Natural language helps to align the user's understanding of the requirements with the analyst's own viewpoint. What the requirements originator really meant and the "facts" that can be derived from the statements that were given are not always congruent [1]. Incomplete and faulty requirements emerge not only during the elicitation of requirements, but also during requirements analysis and modeling [2]. Requirement specifications often need to be revised and rewritten during one of the many iterations in the software development process. Focusing on the improvement of requirements in the first stages of the software development process saves time and improves the final results of a software [3]. So far, this is an inter-disciplinary and human centered process. Improving and accelerating the manual processes with machine support is desirable.

This paper represents the concept of a machine supported tool which helps the analyst to improve natural language specification during the authoring process. We present as proof of concept that the automatic creation of UML models from natural language specifications can be enhanced with "common sense" from an ontology. Since natural language will stay the predominant kind of requirements notation for some time [2], [4], Natural Language Processing (NLP) is a valid approach for the requirements engineering process. The deficits

<sup>1</sup>A requirements analyst is a person who knows how to ask questions for different stakeholders, has experience in working with customers and the management of existing requirements, etc. The characteristics a perfect analyst should have are explained in [1].

<sup>2</sup>Requirements Engineering Specification Improver

of NLP are the understanding and processing of semantics and their correlating side effects. Therefore, including ontologies for semantic knowledge integration into the NLP process is an obvious improvement. Every ontology contains a structure of world knowledge (or domain knowledge) which can solve some of the problems in the requirements engineering process as listed in Section V.

We are currently developing a generic solution called RESI to query various ontologies such as WordNet [5], Research-Cyc [6], and ConceptNet [7] in the correct context during NLP. This approach supports the analyst while examining a large number of requirements.

Preliminary results show that ontologies can improve the authoring and elicitation process of natural language requirements specifications [8], [9]. As requirements specifications are mostly huge document bundles which sometimes take days, weeks, or months to examine, a system that supports the analyst during the requirements engineering process could be a real problem solver. This paper describes a tool which can improve natural language requirements during the automatic modeling process.

The following Section II covers the related work. Some problems that occur and need to be solved during the manual requirements elicitation process are explained in Section III via examples. Section IV shows how RESI works and demonstrates how certain issues can be solved. An abstract discussion about well-known problems in requirements engineering is led in Section V. We point out how we plan to answer these problems with RESI. A case study follows in Section VI and the paper ends with the conclusion in Section VII.

## II. RELATED WORK

Requirements engineering is concerned with the elicitation of high level goals. These goals are to be achieved by the envisioned system [10]. Requirements engineering includes the refinement of such goals and their operationalization into specifications of services and constraints. Today, several approaches use domain specific ontologies to cope with the problems that occur in the requirements engineering process [11], [12], [13], [14], [15]. Some of these projects research the application of formal specifications. They use ontology based systems to correctly classify textual information that has

been delivered from stakeholders. Other projects for example make sure that the correct (domain specific) wording is used when the specification documents are elicited from different stakeholders. Some projects have a narrow field of application and are specified for certain conditions and use cases [16], [7]. Until today, real world applications as listed in [17] have not yet adopted many of the research approaches.

In 2000, Nuseibeh and Easterbrook [4] drafted a road map which shows future research areas to cope with the problems in requirements engineering. They enumerate especially the development of new techniques and the bridging of the gap between contextual inquiry and formal representations.

In 2007, Cheng and Atlee [2] wrote a detailed summary about the requirements engineering's state of the art. They categorize the various topics and try to predict the future of requirements engineering research. Since requirements engineering consumes a lot of time during the software engineering process and has long-term effects to every following stage of the development process, focusing on its improvement is desirable.

Complex and time consuming manual tasks have not yet pervasive tool support [2]. Automating parts of this procedure could not only speed up the processing times of requirements but also decrease error rates.

Cheng and Atlee conclude that it is important to realize that requirements define the problem, not the software itself. They show that requirements engineering activities – in contrast to other software engineering activities – are more iterative, involve more players who have more varied backgrounds and expertise, require more extensive analyses of options, and call for more complicated verifications of more diverse components (e.g. software, hardware, human).

As a possible solution, formal representations of natural language formalize the semantics of statements. This is impossible without a loss of usability or information. Formal specification languages are often perceived as difficult to use by practitioners [18]. Formal approaches need especially trained and skilled analysts to formalize natural language requirements [19]. Also after the formalization, it is hard for stakeholders to take part in further discussions about the requirements. They are not trained to think, act, and talk in formal representations like predicate logic or similar techniques [20]. So far, there is almost no evaluation of how well requirements engineering research reflects in industrial applications.

On the other hand, Fantechi et al. [21] explain that natural language is the perfect vehicle to represent use cases. No other language is as expressive as natural language. The use of natural language is also encouraging to end users who can easily follow and validate the use cases.

Robinson and Pawlowski [22] present empirical studies that show the difficulties and communication breakdowns that requirements engineering processes are frequently experiencing. Requirements inconsistency is a critical driver of the complete requirements engineering process and the requirements dialog.

Therefore, many projects focus on the disambiguation of

natural language specifications [23], [24]. As Kiyavitskaya et al. mention, synonyms in specifications are mostly detected through the human analyst's domain knowledge [25].

### III. EXAMPLES OF ONTOLOGY IMPROVEMENTS

Domain and world knowledge help humans to process requirements, but understanding requirements is still complex. Problems occur not only when there is a lack of domain specific knowledge, but also when special knowledge and expertise overlap. Then stakeholders misunderstand each other without even noticing.

#### A. The Meaning of a Sentence

Consider the following sentence for example:

**The gain should be doubled.**

This sentence includes three major problems within five words.

1) *Exceptional Case*: The modal operation does not cover the exceptional case. The sentence does not describe what needs to be done if the gain is not doubled.

2) *Homonyms and Polysemy*: The word `gain` does have several meanings. Depending on the context and the stakeholders involved in the requirements elicitation process, it could mean gain as in “financial revenue” (this is how an MBA would interpret it) or gain as in “tube voltage” (as an electric engineer would interpret the sentence). This is called lexical ambiguity. Depending on background and education, the analyst might realize the knowledge gap and the fact that `gain` does mean something entirely different in this case. But what happens if not? It is important to distinguish the use of such words and realize their meaning.

3) *Quantity*: The sentence brings in the quantity `double`. It is not correctly quantified since we cannot decide whether it means *exactly* 2.0, that is a 100% increase. Maybe `double` also applies when next quarters revenue numbers reach anything from 180% to 220% of the comparison value.

#### B. Disambiguation with Semantics

A short example:

(1) Tom saw the plane flying.  
(2) Tom saw the mountains flying.

Sentence (1) states the fact that the plane is flying while being seen from a spectator on the ground. Sentence (2) resembles exactly the same structure as its predecessor. The sentence bears several ambiguities which need to be resolved when working with requirements. Ontology knowledge can enhance this process by suggesting the user which aspects are more relevant and by reformatting the sentence. In the first sentence, the object `plane` is flying and the subject `Tom` seeing. In the 2nd sentence, the verb `flying` (most likely) references the subject `Tom` and not the object `mountains`. In this case, distinguishing the difference using parsers does not work. Instead, ontologies know that mountains do not fly, but planes do.

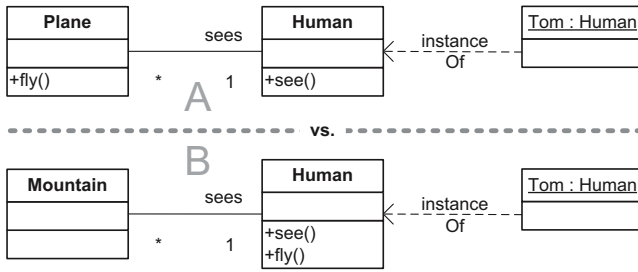


Fig. 1. Class Diagram

When modeling the above sentences in UML, the resulting class diagram of the first sentence looks like Figure 1A compared to the 2nd sentence in Figure 1B. Finding the meaning of words in their sentence structure is vital to software projects, especially when the analyst is not a domain expert. This is often the case, especially in times of offshoring.

### C. Types of Ambiguity

As listed in Ceccato [26], there are many different types of ambiguities: Syntactic ambiguity occurs when grammatical structures lead to a different meaning of a sentence.

**Porcelain egg container.**

Does this describe an egg made out of porcelain, or is it the container that is made from porcelain?

Semantic ambiguity occurs when a sentence has more than one way of reading it. Take the the following sentences for example.

- (1) **Every man loves a woman.**
- (2) **I saw a man on the hill with a telescope.**

Does sentence (1) mean that all men love the very same woman or that each man loves “his” woman. And who carries the telescope in sentence (2)? Who is standing on the hill? There are many possible meanings (permutations) for each of the above sentences.

### D. Coherence Checking

Other problems arise with references that stem from context knowledge. Checking coherence for example looks like this:

**Tom is a man.  
Larry is a cat.  
He lifts him up and puts him in his basket.**

The meaning of who he is becomes instantly clear to a human observer. This is due to the background information humans gather from their own world ontology. A query to an ontology such as ResearchCyc [6] would result in the information that human beings weigh anything between 10 to 250 times as much as regular domestic cats. Therefore it is not possible for the cat to lift the human, but vice versa. Of course the supporting system would have to ask the user to specify what kind of cat the text is referencing to. It makes a number of suggestions about the most likely case from what it already knows from the ontology. If the cat is a tiger, this might result in a different interpretation.

TABLE I  
THEMATIC ROLES AND THEIR MEANING

Thematic Role	Explanation
AG	The acting person or thing executing the action.
ACT	The action, executed by person or thing.
PAT	Person or thing affected by the action or on which action is being performed.
HAB	Possession or belonging; person or thing being received or passed on by person or thing.
POSS	The (current) owner of an element. The “possessor”.

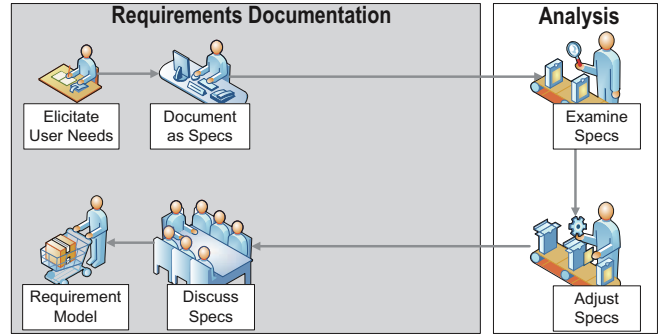


Fig. 2. Manual Requirements Engineering Process

## IV. THE RESI PROCEDURE

As shown in related work, it is important to “formalize the process without formalizing it”. This enables users to still understand the specification while machines can process the input. We use semantic annotation to enrich the specification with additional information [8], [9].

### A. Formalizing Textual Specifications through Annotation

Semantic annotation is done by using thematic roles [9]. Thematic roles describe the role of each element<sup>3</sup> in a phrase. A short excerpt of the 70 thematic roles [27] we are working with can be found in Table I. This additional semantic information can also be used to query the ontology with its domain or world knowledge. The results support the analyst during the requirement creation process.

The manual process depicted in Figure 2 shows the stages of a requirements documentation process as it is today: First the user needs are elicited. Then they are converted into requirements, documented as a specification, and later examined. The analysis most likely leads to adjustments in the requirements and their documentation. The process is iterative and after the requirements pass the quality gate, they go back to stakeholders where they need to be discussed and approved. These requirements form the basis of the requirement models which are passed on to other branches involved (e.g. development, quality assurance).

<sup>3</sup>In linguistics, this is called a constituent. A constituent is a word or a group of words that functions as a single unit within a hierarchical structure.

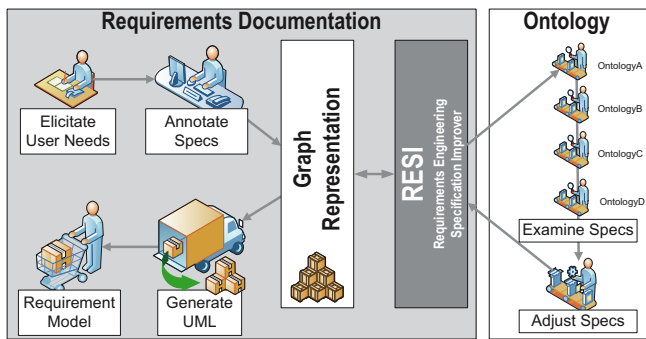


Fig. 3. Requirements Engineering supported by Ontologies

The new improved process that RESI uses is shown in Figure 3. It is semi-automated and uses the text as well as the semantic annotation of the specification. The semantic annotation leads to an altered specifications which is enriched with additional information. The specification is then compiled into a graph system which makes the specification machine readable. RESI has an interface to the graph system and is able to communicate bidirectionally.

### B. Ontology as Magic Box

We are currently implementing RESI to run natural language queries on several ontologies simultaneously (see Figure 3). Depending on whether a domain specific knowledge base or a certain lexical ontology is necessary to retrieve the corresponding information, RESI will query the respective sources and take the corresponding action. For example, this could be a change in the sentence structure or a replacement of a certain term or subject. Depending on the type of the results, it is possible that the system cannot decide which action to take. User interaction via a graphical user interface is necessary. A collection of the type of queries that can be solved with RESI is listed in Section V.

### C. Closing the Loop

The results RESI gathers from the different ontologies lead to changes in the graph. These improvements are fed back to the graph system (see Figure 3). The altered graph represents an improved specification which is more likely to support the development team in the software creation process since many standard obstacles of the requirements engineering process have been removed already. After the text has been formally represented in the graph, it is possible to create UML specific XMI files from this graph [28]. This ensures the direct connection between the natural language specifications and their model representations.

## V. CHALLENGES AND SOLUTIONS

There are many challenges when supporting the human decision making with ontologies. According to Rupp [29], the most important problems with natural language specifications lie in:

### A. Nominalization

Processes are sometimes hidden in a nominalization which needs to be explicitly specified for a correct requirement. E.g. the noun *notification* from the verb *notify*. This nominalization could be replaced by the explicit phrase *send message from A to B which includes information XYZ*.

### B. Incomplete Process Words

Process words (such as verbs) need to be written in active voice. If passive voice is used, an actor (thematic role AG, see Table I) needs to be specified. Also all participants and circumstances involved in the process need to be stated. RESI checks the valency and ensures that all possible configurations are given indeed. It checks that n-ary predicates have all their n references.

### C. Nouns without Reference Index

Not only process words, but also nouns need to be referenced and specified completely so that they do not represent super groups or sub groups of certain sets. The focus is especially on the articles *a* and *the* which are replaced with specific declarations like *every*, *five*, *nobody*, *none*, ... after involving the user via the graphical interface.

### D. Incomplete Specified Conditions

When using conditional branching, the system needs to make sure that every condition does have its full set of possible branches to consider. It needs to check that every if-branch has an else-branch. Correctly specifying these conditions eliminates the problem of implicit knowledge which could lead to fatal assumptions and errors.

### E. Modal Operators Expressing Necessity

Modal operators express conditions which ought to happen. These condition must not appear without the definition what to do in the exceptional case when the desired behavior cannot be met.

### F. Implicit Assumptions (Presumptions)

Conducting the requirements elicitation and engineering process often confronts the analyst with the *curse of knowledge*. It leads to implicit assumptions on requirements and their aspects which are followed by misinterpretations. Finally, it could lead to omitting valuable information during the requirements description. RESI needs to make sure that relations between objects are adequately specified. For example: Every property (thematic role *HAB*) must have an owner (thematic role *POSS*) assigned.

All the above challenges can be addressed by RESI. For evaluation purposes, the system would also have to fulfill other functions. Since the system makes changes on the natural language specifications itself (or its model representation, to be precise), the changes need to be traceable. The specification needs to be legible after the improvements to the model have been made. This ensures

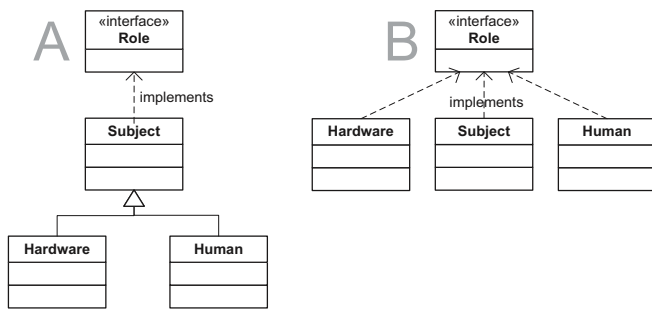


Fig. 4. Partial UML Representation of OMG’s “Actor”

the usability for the stakeholders. The queries deliver results from various ontology sources to one specific problem. The results of these queries need to be combined and presented to the analyst in a fast and easy way. This turns into a problem if the results contradict each other. When using ontology systems with different degrees of generality, this behavior is not uncommon. A meaningful and sensible set of parameters for the treatment of queries and their results is required.

## VI. CASE STUDY

This section shows examples from real world specifications. For an evaluation of the concept proposed in this paper, we took several freely available software specifications and discussed them in a group of requirements analysts. We realized that there are various viewpoints to certain aspects. We argued about the meaning of the terms in the sentences and monitored the decision making mechanism used in our discussions. After the specifications had been reviewed, we checked our list of decisions and compared them to the information we retrieve from various ontologies. By applying the correct queries, many question can be answered with ontologies. Exemplary abstracts of the used specifications are listed below.

### A. Object Management Group UML Specification

The Object Management Group’s UML specification [30] describes the figure “Actor” in the class description of use cases (see paragraph 16.3.1) as the following:

- (1) **An Actor models a type of role played by an entity that interacts with the subject [...], but which is external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject).**
- (2) **Actors may represent roles played by human users, external hardware, or other subjects.**

The first sentence has the verb *interact*. It is not clear whether it is *type of role* that interacts or the *entity*. RESI asks the user which role to denote to the corresponding objects. The second sentence mentions the other subjects. It is not clear whether human users and external hardware are also subjects. If they are, Figure 4A would be a correct UML representation. Otherwise Figure 4B could be a correct solution. The user has to be asked which deduction is feasible.

### B. W3C HTML Specification

The W3C<sup>4</sup> states the following in the HTML specification [31]:

- (1) **An HTML form is a section of a document containing normal content, markup, special elements called controls (checkboxes, radio buttons, menus, etc.), and labels on those controls.**
- (2) **Users generally "complete" a form by modifying its controls (entering text, selecting menu items, etc.), before submitting the form to an agent for processing (e.g., to a Web server, to a mail server, etc.)**
- (3) **Users interact with forms through named controls.**

The verb *contain* in sentence (1) could reference section or document. When reading this in a specification, RESI has to ask which reference to take. The specification has a quote “complete”. Its preposition *by* denotes that the term *modifying its controls* specifies the term *complete* in more detail. Quotes are non-specific and ought not to be used for the sake of precision in sentences. RESI points out this flaw to the user.

### C. Ludo Specification

The textual specification of the game Ludo can be found in [32]. We list a short excerpt of the specification here:

- (1) **There are four players in a cyclic order: red, blue, yellow, and green.**
- (2) **If one of the following moves is possible, the player must choose one**

Humans realize that the colors in sentence (1) could be modeled as a single attribute (e.g. *color*) with an enumeration of values (*red, blue, yellow, green*). Ontologies know colors. Therefore RESI does not model four different attributes but recommends to group these attributes. The approach is obvious in this case, but becomes a lot more valuable once the analyst is not a domain expert. This disadvantage could be alleviated with a system that reads ahead and hints to conceptual relatedness as shown in [8].

The second sentence states *If one [...] is possible* and shows a familiar problem that most specifications possess: incomplete conditions. We worked out the details of incomplete conditions in Section IV-C and Section V-D. RESI should raise the question “What happens if none of the moves are possible?”.

## VII. CONCLUSION

The area of requirements engineering is important to any project. Until today, research focuses on improving the error-prone and complex manual processes that humans have to carry out. The elicitation of requirements and the review of the results still lie in the hands of the analyst.

Software systems have huge requirement collections that span hundreds if not thousands of pages. The probability that

<sup>4</sup>World Wide Web Consortium

facts are overlooked, misunderstood, or not completely specified is high. Many projects tend to go over time and budget because requirement specifications are faulty or incomplete. Most of the times these errors emerge at a later stage in the software development process and lead to immense costs for change requests. Formal approaches try to deliver a solution, but turn out to be quite complex to use.

User adaption is very important: all kinds of stakeholders with different levels of background knowledge need to interact during the requirements elicitation and engineering. Requirements are the interface between the expert and the customer. If we want to make sure that the customer can understand the requirements, we need to provide textual specifications which can be revised by the customer yet provide the requirements analyst and the development team with the necessary information for their work. Speeding up and improving this procedure by using ontology based recommender systems such as RESI seems a logical consequence. Our studies have shown that this solution is feasible and we are already implementing the software system that addresses the issues listed in this paper. RESI also supports the analyst by asking questions which limits the risk of overseeing aspects of the specification.

In the future, research in artificial intelligence will improve the algorithms for deduction as well as the speed and coverage of ontologies. The number of semantic applications increases and storage space and memory needed for the successful application of ontologies and their functions become more affordable.

#### ACKNOWLEDGMENT

AUTOMODEL is funded by ec4u expert consulting ag, Germany in cooperation with the University of Karlsruhe, Germany.

#### REFERENCES

- [1] C. Rupp and R. Goetz, "Psychotherapy for System Requirements," *Proceedings of the Second IEEE International Conference on Cognitive Informatics (ICCI '03)*, 2003.
- [2] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering," in *Proc. Future of Software Engineering FOSE '07*, 23–25 May 2007, pp. 285–303.
- [3] C. Rupp, "Requirements and Psychology," *IEEE, May/June 2002*, vol. IEEE SOFTWARE, 2002.
- [4] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," in *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: ACM Press, 2000, pp. 35–46.
- [5] G. A. Miller, C. Fellbaum, R. Tengi, P. Wakefield, H. Langone, and B. R. Haskell, "WordNet." [Online]. Available: <http://wordnet.princeton.edu/>
- [6] Cycorp Inc., *Cyc / ResearchCyc*, Cyc.com. [Online]. Available: <http://cyc.com/>
- [7] C. Havasi, R. Speer, and J. Alonso, "ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge," in *Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September 2007. [Online]. Available: <http://web.media.mit.edu/~jalonso/cnet3.pdf>
- [8] S. J. Körner and T. Gelhausen, "Improving Automatic Model Creation using Ontologies," in *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering*, Knowledge Systems Institute, Ed., Jul. 2008, pp. 691–696.
- [9] T. Gelhausen and W. F. Tichy, "Thematic Role Based Generation of UML Models from Real World Requirements," in *Proc. International Conference on Semantic Computing ICSC 2007*, 2007, pp. 282–289.

- [10] A. van Lamsweerde, R. Darimont, and E. Letier, "Managing conflicts in goal-driven requirements engineering," vol. 24, no. 11, pp. 908–926, Nov. 1998.
- [11] H. Kaiya and M. Saeki, "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach," in *Proc. Fifth International Conference on Quality Software (QSIC 2005)*, 19–20 Sept. 2005, pp. 223–230.
- [12] —, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," in *Proc. th IEEE International Conference Requirements Engineering*, 11–15 Sept. 2006, pp. 189–198.
- [13] M. Saeki, "Ontology-Based Software Development Techniques," *ERCIM News*, vol. 58, pp. 14–15, 2004.
- [14] Y. Zhang, R. Witte, J. Rilling, and V. Haarslev, "An ontology-based approach for traceability recovery," 2006.
- [15] W. Meng, J. Rilling, Y. Zhang, R. Witte, and P. Charland, "An Ontological Software Comprehension Process Model," *3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM 2006)*. October 1st, Genoa, Italy, 2006.
- [16] H. Liu and P. Singh, "ConceptNet - a practical commonsense reasoning tool-kit," *BT Technology Journal*, vol. Vol 22, 2004. [Online]. Available: <http://larifari.org/writing/BTTT2004-ConceptNet.pdf>
- [17] Volere, "List of requirement engineering tools," 2009. [Online]. Available: <http://www.volere.co.uk/tools.htm>
- [18] S. Konrad and B. H. Cheng, "Facilitating the Construction of Specification Pattern-based Properties," *IEEE International Conference on Requirements Engineering*, pp. 329–338, 2005.
- [19] A. Pease and W. Murray, "An English to Logic Translator for Ontology-based Knowledge Representation Languages," *IEEE 0-7803-7902-0/03*, pp. 777–783, 2003.
- [20] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw, "Automated Consistency Checking of Requirements Specifications," *ACM Trans. Softw. Eng. Methodol.*, vol. 5, no. 3, pp. 231–261, 1996.
- [21] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, "Application of Linguistic Techniques for Use Case Analysis," *IEEE International Conference on Requirements Engineering*, p. 157, 2002.
- [22] W. N. Robinson and S. D. Pawlowski, "Managing requirements inconsistency with development goal monitors," vol. 25, no. 6, pp. 816–835, Nov.–Dec. 1999. [Online]. Available: <http://www.cis.gsu.edu/~wrobinso/papers/TSE99.PDF>
- [23] L. Goldin and D. M. Berry, "AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation," *Automated Software Engg.*, vol. 4, no. 4, pp. 375–412, 1997.
- [24] C. Denger, D. M. Berry, and E. Kamsties, "Higher Quality Requirements Specifications through Natural Language Patterns." Los Alamitos, CA, USA: IEEE Computer Society, 2003, p. 80.
- [25] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," *Requir. Eng.*, vol. 13, no. 3, pp. 207–239, 2008.
- [26] M. Ceccato, N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Ambiguity Identification and Measurement in Natural Language Texts." [Online]. Available: <http://eprints.biblio.unitn.it/archive/00000727/>
- [27] T. Gelhausen, B. Derre, M. Landhäußer, T. Brumm, and S. J. Körner, "SaleMX Project Website - the Thematic Roles of SALE," 2008. [Online]. Available: <http://svn.ipd.uni-karlsruhe.de/trac/mx/wiki/MX/SALE/ThematicRoles>
- [28] T. Gelhausen, B. Derre, and R. Geiss, "Customizing GrGen.NET for Model Transformation," in *GRAMoT '08: Proceedings of the 3rd International Workshop on Graph and Model Transformation*. Leipzig, Germany: ACM, May 2008, pp. 17–24. [Online]. Available: <http://www.ipd.uka.de/Tichy/uploads/publikationen/180/gramot2-gelhausen.pdf>
- [29] Chris Rupp & die SOPHISTen, *Requirements-Engineering und Management*, 4th ed. Carl Hanser Verlag, 2006.
- [30] Object Management Group, "Unified Modeling Language: Superstructure Version 2.1.1 16.3 Class Description 16.3.1 Actor," PDF, February 2007. [Online]. Available: <http://www.omg.org/docs/formal/07-02-03.pdf>
- [31] W3C, "HTML 4.01 Specification, 17 Forms," December 1999. [Online]. Available: <http://www.w3.org/TR/html401/interact/forms.html>
- [32] M. Kroll and R. Geiss, "A Ludo Board Game for the AGTIVE 2007 Tool Contest," 2007. [Online]. Available: <http://www.informatik.uni-marburg.de/~swt/active-contest/Ludo-Karlsruhe.pdf>

# A Knowledge-Based Retrieval Model

Fabio Silva, Rosario Girardi, and Lucas Drumond  
UFMA – Federal University of Maranhão  
Avenida dos Portugueses, s/n, São Luís, Brazil

## Abstract

*Several techniques for extracting meaning from text in order to construct more accurate internal representations of both queries and information items in retrieval systems have been already proposed. However, there is a lack of semantic retrieval models to provide appropriate abstractions of these techniques. This article proposes a knowledge-based information retrieval model that explores the semantic content of information items. The internal representation of information items is based on user interest groups, called "semantic cases". The model also defines a similarity measure for ordering the results based on the semantic distance between semantic cases items. An initial experiment evaluating the model is also described.*

## 1. Introduction

The effectiveness of keyword-based processes models is limited by the phenomenon known as "keyword barrier", i.e. the internal representation of an information item based on a set of words extracted from texts through statistical and / or syntactic techniques does not allow a considerable improvement of the effectiveness of information retrieval results [6].

These limitations have stimulated the development of several techniques trying to extract meaning from texts, such as semantic analysis, to obtain more accurate internal representations of information items [6][12]. However, there is a lack of semantic retrieval process models providing appropriate abstraction representations of the activities, products and techniques involved in such retrieval processes.

With a semantically structured representation of information items, retrieval systems can use semantic-based techniques in order to improve their effectiveness like document annotation, knowledge-based search, query expansion and ontology-based similarity measures.

This work proposes a retrieval process model which uses ontology-based structures to represent information items

and semantic case-based similarity measures. Upon this model, new systems can be built applying different techniques to find the best configuration for domain retrieval tasks.

The paper is organized as follows. Section 2 formalizes the ontology and knowledge base concepts used in this work. Section 3 introduces the main components of a generic information retrieval model, describing the semantic case strategy and detailing the components of proposed model. Section 4 describes an experiment developed for evaluating the proposed model. Section 5 discusses related work on semantic-based information retrieval systems. Section 6 concludes the article and suggests directions for future work.

## 2. Ontologies

Ontologies are defined as formal specifications of domain conceptualizations [7], providing a shared vocabulary for representing knowledge. In order for the meaning of their expressions being machine readable, ontology languages were created with a formal logic-based semantic. An ontology is defined as the 5-tuple [15]:

$$O := (C, H^C, R, rel, A^O) \quad (1)$$

where:

- $C$  is the set of ontology concepts. Concepts represent entities of a specific domain. They are designated by one or more natural language terms and have also a unique identifier inside the ontology;
- $H^C \subseteq C \times C$  is a set of taxonomic relationships between the concepts. Such taxonomic relationships raise a hierarchical structure, denoted as  $H^C = \{(C_i, C_j) | C_i, C_j \in C \wedge C_i \sqsubseteq C_j\}$ ;
- $R$  is set of non-taxonomic relationships between concepts;
- $rel$  is a function  $rel : R \rightarrow C \times C$  that maps relation identifiers to their respective relations. This can be de-

noted as  $rel(r) = (c_1, c_2)$  or as  $r(c_1, c_2)$  where  $r \in R$  and  $c_1, c_2 \in C$ ;

- $A^O$  is a set of axioms described by a logic language. Axioms can be used for checking ontology consistency and deduce new knowledge from ontology through some inference mechanism.

Ontologies define the vocabulary of an application domain through concepts, a general or intensional knowledge. Specific or extensional knowledge about the concrete world is described through instances of concepts. For instance, a concept *City* represents a general city while the instance *New York* refers to the real American city. A knowledge base which contains these two kinds of knowledge in a single model is defined as the 4-tuple [14]:

$$KB := \{O, I, inst_c, inst_r\} \quad (2)$$

where:

- $O$  is an ontology;
- $I$  is a set of instances;
- $inst_c$  is a function  $inst_C : I \rightarrow C$  that maps the instances to their associated concepts;
- $inst_r$  is a function  $inst_R : I \rightarrow R$  that maps the instances to their associated relations.

### 3. The Proposed Retrieval Model

In this work, we adopted the definition of an information retrieval model provided in [2]. An information retrieval model is a quadruple:

$$(D, Q, \mathcal{F}, R(d, q)) \quad (3)$$

where:

- $D$  and  $Q$  are sets with, respectively, the internal representation of documents and queries;
- $\mathcal{F}$  is a framework for modeling document representations, queries, and their relationships;
- $R(d, q)$  is a ranking function for ordering documents with regard to the query  $q$ .

Documents and queries are represented as ontology instances. The model is composed of a knowledge base, a semantic case-based representation strategy and a similarity measure between concepts. The knowledge base stores the information items and provides an uniform representation for documents and queries since metadata share a common conceptualization. A semantic case-based representation strategy defines how the metadata is organized into the

internal representation of documents. The similarity measure used by the ranking function obtains a numeric value from the semantic distance between concepts. This ranking function also uses a semantic case-based strategy for ordering results.

#### 3.1. Semantic Cases

A semantic case represents a characteristic of an information item through which user interests can be specified [5]. User interests in the proposed model are topics, with each topic being described by an ontology sub-hierarchy. Thus, a topic comprises a general concept, root of the respective sub-hierarchy, and its sub-concepts. Each topic is called a “semantic case”. These general concepts - also called as semantic case root concepts - have characteristics that represent different and independent aspects of the domain. These characteristics are preserved by specializations of the general concepts. These general concepts can be selected with the help of a domain specialist. A semantic case is a pair defined by:

$$S = (C_r, T) \quad (4)$$

where  $C_r$  is the root concept and  $T$  is a set of sub concepts of  $C_r$ ,  $T = \{C_i | C_i \sqsubseteq C_r\}$ .

An information item refers to a semantic case if a sub-concept of the semantic case root concept, also called “semantic case terms”, appears in its internal representation. An example of a semantic case can be seen in Figure 1, where the concepts *Person*, *Content* and *Location* can be considered as general concepts of a domain. For instance, the semantic case created from the root concept *Location* is given by  $S_{Location} = (Location, \{Location, City, Country\})$ .

#### 3.2. Internal Representation of Information Items

The internal representation uses a semantic case-based representation strategy to organize elements, i.e. concepts and instances, present in a document with regards the semantic cases. As defined in section 3.1, an ontology concept,  $c \in C$ , is a term of the semantic case represented by  $S_j$  if  $c \in T_j$ . An internal representation instance,  $i \in I$ , also can be a term of a semantic case  $S_j$  if its most specific concept is a term of the semantic case, i. e.,  $msc(i) \in T_j$ . Once defined the criteria to classify concepts and instances, the internal representation can be obtained from the following definitions:

**Definition 1:** Let  $C_d$  the set of concepts,  $n$  the total of instances and  $I_d$  the set of instances, all present in a document  $d$ . The set of concepts that are referenced by instances is  $C_i = \{msc(i_1), \dots, msc(i_n)\}$ . The set  $CR_d$



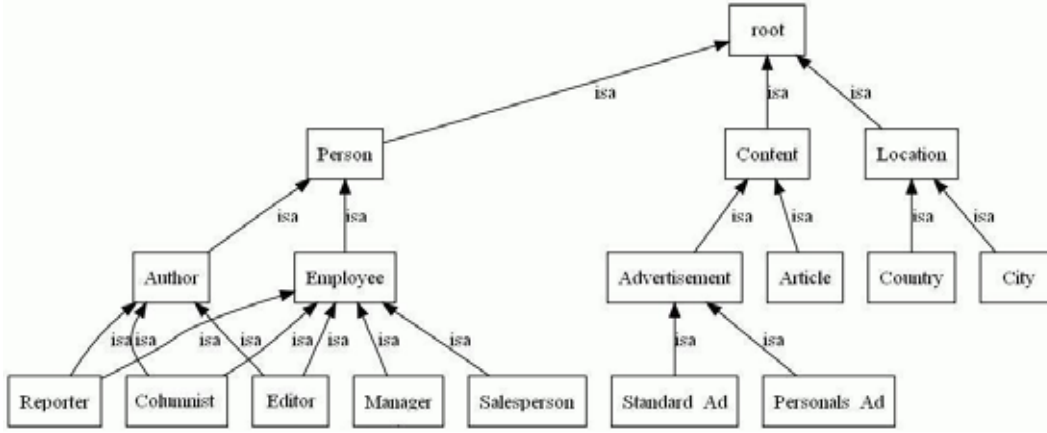


Figure 1 . A newspaper ontology example

of all representative concepts from  $d$  is given by the union of the both set of concepts,  $CR_d = C_d \cup C_i$ .

For example, consider a document containing the concepts *City* and *Article*, and the instance *Thomas Friedman* of the concepts *Author*, *Person* and *Columnist*, in the domain shown in Figure 1. The sets obtained from the definition 1 are  $C_d = \{City, Article\}$  and  $C_i = \{Columnist\}$ , since  $msc(Thomas\ Friedman) = Columnist$ .

**Definition 2:** Let  $SS = \{S_1, \dots, S_m\}$  the set of semantic cases in a domain. The document  $d_k$  is represented by a set of pairs, each pair  $P$  associated with a semantic case,  $d_k = \{P_{k1}, \dots, P_{km}\}$ . The pair  $P_{kj}$  contains the set of representative concepts and the set of instances present in  $d_k$  associated to the semantic case  $S_j$ ,  $P_{kj} = (CR_{kj}, I_{kj})$ .

The internal representation of a document according to the definition 2, is given by:

$$d_k = \{(CR_{k1}, I_{k1}), \dots, (CR_{km}, I_{km})\}$$

For example, consider the sentence "Thomas Friedman wrote an article about New York..." and the ontology shown in Figure 1. The elements extracted from the sentence are shown in the Table 1.

Table 1 . Example of the internal representation of an item

Components	Values
Instance	<i>Thomas Friedman</i>
Instance	<i>New York</i>
Concept	<i>Article</i>
Concept	$msc(Thomas\ Friedman)$
Concept	$msc(New\ York)$

The elements are arranged in pairs with respect their se-

mantic case, in the example domain: *Person*, *Content* or *Location*. The internal representation of the document is the following set:

$$d = \{(\{Columnist\}, \{Thomas\ Friedman\}), (\{Article\}, \{\}), (\{City\}, \{New\ York\})\}$$

### 3.3. Matching Process

In an ontology context, relevant documents can refer to super- and subconcepts of the searched concepts. The set of all super- and subconcepts of a given concept  $C_i$  is called "semantic cotopy" of  $C_i$  [15], defined as:

$$SC(C_i, H^C) = \{C_j \in C | H^C(C_j, C_i) \vee H^C(C_i, C_j)\} \quad (5)$$

The matching strategy uses the concepts in the internal representation of document. In order to decrease the cognitive distance between a query and an information need, the retrieval function selects concepts appearing in the documents with similar semantic content of query concepts. In other words, document concepts must belong to a "semantic cotopy" of query concepts.

The retrieval function is given by:

$$RF(q, d) := \begin{cases} 1, & \text{if } \exists c_i c_j | \\ & (c_i \in CR_q) \wedge (c_j \in CR_d) \wedge (c_j \in SC(c_i, H^C)) \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where  $CR_q$  and  $CR_d$  are sets of concepts in the query and document pairs.

Consider as an example the following user query  $q$ : *authors writing about New York*, which internal representation is:

$$d = \{(\{Author\}, \{\}), (\{\}, \{\}), (\{City\}, \{New\ York\})\}$$

When trying to compute the retrieval function for  $q$  and the document of Table 1, called  $d$ , one must search for the concepts in  $d$  (i.e. *Columnist*, *City* and *Article*) that are also in the semantic cotopy of the concepts in  $q$  (i.e. *Author* and *City*). The “semantic cotopy” of each concept in the query is the following:

- $SC(Author, H^C) = \{Root, Person, Author, Reporter, Editor, Columnist\}$ ;
- $SC(City, H^C) = \{Root, Location, City\}$ .

The concepts *Columnist* and *City* appear both in the semantic cotopy of the concepts in  $q$  and in the internal representation of  $d$ . Therefore the value  $RF(q, d)$  is 1.

### 3.4. Similarity Analysis

In an information retrieval model, similarity analysis estimates the relevance of each document regarding a given query. The matching strategy described above ensures that retrieved items always have some relevance. Thus, the proposed similarity model aims at defining an ordering for the retrieved documents. This ordering is represented by the function  $R(d, q)$  as stated in the generic retrieval model description.

In the proposed model, the internal representation of an information item  $d$  is a set of pairs, according to definition 2. In order to determine the relevance of a document for a given query, comparison criteria for the pairs in the query and in the document representation are needed. The proposed similarity analysis uses an approach similar to the one in [5][6]. In these approaches, possible groups of interest in a domain are represented as semantic cases. The proposed similarity function matches the pairs in the document with the ones in the query that refer to the same semantic case, i.e. that have the same semantic case index  $i$ . The elements of the sets  $CR_i$  and  $I_i$  of the matching pairs are compared according to some semantic similarity measure. The relevance of the document is the normalized sum of the similarity between the matching pairs. Additionally, a weight  $\omega$  can be associated with each semantic case meaning its relevance for the user query. The similarity function is defined as:

$$R(d, q) = \frac{1}{\sum_{i=1}^m \omega_i} * \sum_{i=1}^m \omega_i * \frac{Sim(CR_{i_d}, CR_{i_q}) + Sim(I_{i_d}, I_{i_q})}{N_{CR_{i_q}} + N_{I_{i_q}}} \quad (7)$$

where:

- $i$  is the semantic case index;
- $m$  is the total of semantic cases;

- $CR_{i_d}$  and  $I_{i_d}$  are sets of concepts and instances from internal representation of the document with regards a semantic case  $i$ ;
- $CR_{i_q}$  and  $I_{i_q}$  are the corresponding sets of the query;
- $N_{CR_{i_q}}$  and  $N_{I_{i_q}}$  are, respectively, the number of concepts and instances extracted from query with regards a semantic case  $i$ .

The similarity between concepts sets in equation 7 is given by:

$$Sim(CR_d, CR_q) = \sum_{j \in CR_q} \max Sim_c(c_{q_j}, c_{d_k}) \quad (8)$$

where:

- $c_{q_j}$  is the element  $j$  of the set  $CR_q$ ;
- $c_{d_k}$  is the element  $k$  of the set  $CR_d$ ;
- $Sim_c$  is a similarity measure between two concepts.
- $max$  obtains the maximum similarity value between the element indexed by  $j$ , computed over all elements of  $CR_d$ .

The similarity between instances sets in equation 7 is given by a function similar to the one in the equation 8.

The similarity model uses a similarity measure between concepts/instances from the sets present in the internal representation of the query and the document. Such similarity measure is not described or suggested here. Several measures for this purpose have already been proposed in the literature [5][13][16], and can be adapted in an instance of the retrieval model proposed in this work in order to compare elements of the sets.

## 4. Evaluation

A preliminary evaluation of the proposed retrieval model effectiveness was performed by instantiating it in a knowledge-based retrieval system and by comparing it with a retrieval system based on the vector space model created using the Apache Lucene <sup>1</sup> API. The system was developed in the tax law domain, using a document collection composed by normative instruments instantiated in a semi-automatic way. The system knowledge base was built through the instantiation of these instruments in the ontologies ONTOJURIS and ONTOTRIB [1]. ONTOJURIS classifies the different legal branches and describes the structure of normative instruments. ONTOTRIB represents concepts of the tax legal branch, where were found the following semantic cases: *tributary normative instrument*

<sup>1</sup><http://lucene.apache.org/>

that maps the user interests in specific instruments, *tribute* that describes all tributes and taxes, *juridical application* that determining the tribute scope (federal, state or municipal) and *tributary concepts* that describe the elements of a juridical-tributary relationship. The similarity measure used was proposed in [5] extended in this work for calculating the similarity between instances of the knowledge base. Identical weights for semantic cases ( $\omega = 1$ ) were considered in the experiment. 8 queries, proposed by a domain specialist, and a pre-defined set of results for each query were specified. The precision values, for different recall measures, were computed and the results are shown in Figure 2.

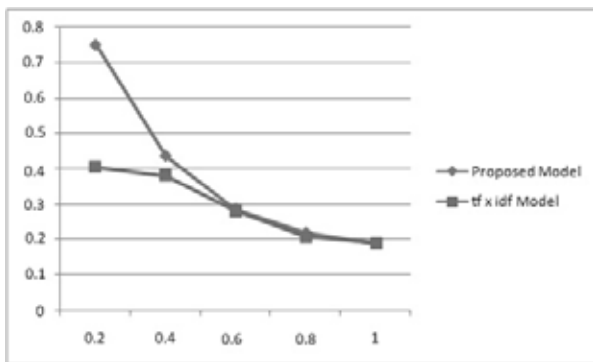


Figure 2 . Comparative graph of recall x precision

The conducted experiment revealed that the system based on the proposed model achieved higher precision rates between 20% and 40% of recall values than the system based on the vector space model. The retrieval system based on the proposed model achieved high effectiveness in the selected domain. The experiment has been performed with a small set of queries and information items because, currently, the internal representations of queries and information items are just semi-automatically created. For an exhaustive evaluation of the proposed model, internal representations should be constructed automatically through ontology population techniques, which is our next research aim.

## 5. Related Work

SIM-DL [10] presents a similarity measure for description logic concepts and uses a discourse context, based on subsumption reasoning, to select comparable concepts. In despite of the semantic cases are pre-selected in the model proposed here, the definition of context concepts occurs in a similar way in both works. However, the proposed model here defines steps for information retrieval in documents, including the matching process, while the SIM-DL framework focuses on the comparison of concepts.

Other Semantic Web search systems have been created

using different strategies from our approach. The possibility of processing formal queries is explored in several of these systems. One of the first proposals for an information retrieval system based on metadata was SHOE [9]. In this system, queries are created through a form with KB elements generating a formal query. Vallet et al [19] presented an ontology-based information retrieval model which takes as input a formal RDQL query for instance retrieval in KB. However, none of these systems provides a similarity measure based on the semantic content of documents.

The metadata obtained from the document annotation process are also used for query expansion. DOSE [3] is a system that evaluates the results and, based on heuristics such as taxonomic relationships, extracts new elements of the knowledge base for query expansion. KIM [11] is a system that focuses the process of creation of metadata from documents. The retrieval process performs full-text indexing and search for terms and metadata. OWLIR [18] uses a representation mechanism and similarity analysis that combines keyword with structured data. In this second group of systems, document retrieval is still based on the vector space model.

Finally, some systems exploit RDF graphs created from knowledge bases. The existing relationships lead to a semantic network. TAP [8] retrieves an initial set of nodes from search terms. The query is expanded by specifying the desired properties and exploiting the sub graphs of the initial set of nodes. Rocha et al [17] propose a semantic network in which relations have both a semantic label and numerical weight. Query terms are mapped to semantic network nodes. A spread activation algorithm is used to discover concepts closely related to the initial concepts. Beagle++ [4] maintains a RDF graph with weights assigned manually to the concepts links in the pre-existing knowledge base. The result combines keyword frequency ranking and an object algorithm ranking. These systems provide additional relevant data extracted from metadata.

## 6. Conclusion and Further Work

This article described a knowledge-based information retrieval model built using semantic components and services like ontologies and inference rules.

The proposed model uses a semantic case strategy for organizing concepts and instances into the internal representation of both queries and documents with respect to a pre-defined information context. This context is considered into retrieval and similarity analysis stages in order to apply semantic similarity measures. The model avoids noisy results using a semantic representation, i.e. concepts and ontology instances, rather than keywords, for representing information items. Queries are expanded using ontological inferences, allowing new documents to be discovered.

Finally, the context defined by semantic cases provides a way to specify and quantify user interests in a ontology sub-hierarchy.

Future work should explore ways to enrich the internal representation through others inference types such as transitivity and inverse relationships. The model application also depends on metadata from documents annotation. More research work for the evaluation of the proposed model is needed through its instantiation with various corpus using ontology population techniques.

## References

- [1] I. Araujo, L. Drumond, R. Mariano, and R. Girardi. ONTOJURIS e ONTOTRIB: ontologias para a modelagem do conhecimento jurídico. *SEMINÁRIO DE PESQUISA EM ONTOLOGIA NO BRASIL UFF, IACS - Departamento de Ciência da Informação, Niterói 08/2008*, Disponível em <http://www.uff.br/ontologia/artigos/314.pdf>. Acessado em 27/10/2008.
- [2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.
- [3] D. Bonino, F. Corno, and F. Farinetti. Dose: a distributed open semantic elaboration platform. *IC-TAI'03*, page 580, 2004.
- [4] P. Chirita, S. Costache, W. Nejdl, and R. Paiu. Beagle++: Semantically enhanced searching and ranking on the desktop. *The Semantic Web: Research and Applications*, 2006.
- [5] L. Drumond, R. Girardi, and F. Silva. A similarity analysis model for Semantic Web information filtering applications. In *Proceedings of the Twentieth International Conference on Software Engineering and Knowledge Engineering (SEKE 2008)*, pages 638–642, Redwood City, California, USA, 2008. Ed. Knowledge Systems Institute Graduate School.
- [6] R. Girardi and B. Ibrahim. Using English to Retrieve Software. *The Journal of Systems and Software*, volume 30, No. 3:pages 249–270, 1995.
- [7] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, volume 43 Issue 5-6:pages 907–928, 1995.
- [8] R. Guha, R. McCool, and E. Miller. Semantic search. In *WWW '03: Proceedings of the 12th int. conf. on World Wide Web*, 2003.
- [9] J. Heflin and J. Hendler. Searching the web with shoe. *Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01*, 2000.
- [10] K. Janowicz. Sim-DL: Towards a semantic similarity measurement theory for the description logic alcnr in geographic information retrieval. In *SeBGIS 2006, OTM Workshops 2006, ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, P. Herrero, and e. al., Eds, Springer*, volume 4278:pages 1681–1692, 2006.
- [11] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics: Science, Services and Agents on the WorldWideWeb*, volume 2:pages 49–79, 2004.
- [12] C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. In Ed. C. Fellbaum, editor, *WordNet: A Lexical Reference System and its Application*, pages 265–283, Cambridge, MA: MIT, 1998.
- [13] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning (ICML) Morgan Kaufman, San Francisco*, pages 296–304, 1998.
- [14] A. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishing, 2002.
- [15] A. Maedche and S. Staab. Measuring similarity between ontologies. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, pages 251–263, 2002.
- [16] P. Resnick. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, pages 95–130, 1999.
- [17] C. Rocha, D. Schwabe, and M. Aragao. A hybrid approach for searching in the Semantic Web. In *Proceedings of the 13th int. conf. on World Wide Web (WWW)*, pages 374–383, 2004.
- [18] U. Shah, T. Finin, and A. Joshi. Information retrieval on the Semantic Web. In *CIKM '02: Proc. Of the 11th int. conf. on Information and knowledge management*, pages 461–468, 2002.
- [19] D. Vallet, M. Fernández, and P. Castells. An ontology-based information retrieval model. *Proc. Second European Semantic Web Conf. (ESWC '05)*, pages 455–470, 2005.

# TRiple Content-Based OnTology (TRICOt) for XML Dissemination

Mirella M. Moro<sup>1</sup>, Deise de Brum Saccol<sup>2</sup>, Renata de Matos Galante<sup>3</sup>

<sup>1</sup> Depto Ciência da Computação, UFMG, Brazil

<sup>2</sup> Universidade Federal do Pampa, Brazil

<sup>3</sup> Instituto de Informática, UFRGS, Brazil

mirella@dcc.ufmg.br, deisesaccol@gmail.com, galante@inf.ufrgs.br

## Abstract

*As Internet and distributed systems evolve, content dissemination systems become a hot topic for researchers in those areas. In such systems, users define profiles (queries) that must be evaluated over incoming messages (documents), usually on streams. Given the high number of profiles and the considerable flow of incoming messages on such systems, research problems reach new levels of complexity on databases and software engineering perspectives as well. For example, those features make distributed query evaluation even more complex. In this context, we propose to expand the use of ontologies to this new context of stream processing. Our initial evaluation shows that such solution is viable and opens new possibilities for using the whole potential of ontologies in a very diverse set of applications.*

## 1. Introduction

In traditional request/reply systems, users submit a query to the system, which evaluates it over the stored data and returns the results back to the users, individually. As Internet and distributed systems evolve, a new paradigm aggregates the concept of *content dissemination* to such applications. Besides answering to the queries considering the stored data, the system also keeps those queries stored. Then, the system evaluates those queries every time new data is added through the application. In this case, the results are *disseminated* to the users *a posteriori*.

This form of querying is widely employed in *content-based information dissemination services*, or simply content dissemination. Such approach can also be instantiated as *publish-subscribe* services. This new paradigm of evaluating data has created opportunities for new types of applications. Such applications are used by notification systems to inform users about new products on the market, updates on stock values, currency variation, and deal offers, for ex-

ample. Also, as Web services spread all over the Internet, new dissemination applications are frequently created.

Since XML is recognized as the standard for data exchange, content-based dissemination services that are specialized on handling XML data become necessary [6]. In XML-based dissemination services, there is a continuous stream of XML messages (usually, an XML message contains an XML document) from producers to consumers [13]. The message transmission is performed by a sophisticated overlay network composed of content-based routers (*XML routers*). Those routers evaluate profiles over messages (or queries over documents) and forward them to their recipients, such as users or other routers. This task of evaluating profiles over messages is called *message filtering*.

It is important to notice that content-based dissemination services are in constant evolution. New queries are requested and new data are added to the system. The amount of queries being processed easily reaches the thousands. The volume of data in a period of time may surpass the gigabytes. Therefore, one key feature of such a system is *scalability* in terms of amount of queries and volume of data that can be processed at the same time.

Besides scalability, another central feature is the ability to *efficiently* evaluate thousands of query. Note that such query evaluation is in *filtering style*. In other words, it is necessary to identify the queries that are satisfied by incoming documents. Usually, efficiency is achieved by performing multi-query processing.

In addition to query and data processing mechanisms, the network architecture also influences the system performance as a whole [16]. An option is to use peer-to-peer (P2P) networks [17]. Among the possible choices, a P2P network has the advantages of reducing network traffic, minimizing routing depth, and preserving those features in case of network updates and failures.

However, documents from the same application domain are usually *spread* over the P2P network, which can influence the message filtering performance. These shortcomings may be overcome by using ontologies to cluster doc-

uments from the same domain into specific peers [4]. This clustering may also be coordinated with the specification of *super peers* that keep the information about ontologies and the respective documents

**Contributions.** Previous work on XML dissemination systems (e.g. [6, 8, 11, 14]) focus on different aspects of the dissemination. This paper focuses on the aspects of document dissemination, query grouping, and query evaluation on XML content-based dissemination services. These three tasks employ an homogeneous mechanism based on ontologies, called TRICOT (*TRIPLE Content-based OnTology*). This is the *first* time that such an uniform treatment is proposed to an XML dissemination service. While the state-of-the-art work try to improve individual aspects, our work considers a global improvement of the system. Another fundamental aspect is that the query evaluation algorithms may also use information from the ontologies in order to improve their performance. The contributions of this paper are:

- We discuss some recent scenarios that show the versatility of dissemination systems - Section 2. Those scenarios may serve as use cases to TRICOT. They also present important, unique requirements that can be further explored.
- We define an ontology-based approach for grouping and evaluating queries - Section 3. The goal is that a document be evaluated considering only the queries defined in its domain, instead of considering the universe of queries. Such approach improves scalability in terms of number of queries.
- We specify a P2P-based architecture for using as the network topology for XML routers - Section 4. Specifically, the dissemination system clusters documents from the same domain into peers. Such approach solves the scalability problem in terms of volume of processed data.

An initial case study was performed to demonstrate the viability of TRICOT and its advantages. Furthermore, the results also show the potential of our solution for using ontologies in dissemination services over P2P networks - section 5. Finally, section 6 overviews related work, and section 7 presents our concluding remarks.

## 2. Content-based Dissemination

Content-based dissemination systems use an interaction model based on events that work in a reverse way to traditional *request/reply* systems. The communication is started by the data provider, and not by its consumer. The roles of queries and data are reversed. Specifically, in request/reply systems, the queries are processed over the data in order

to identify which parts satisfy the query. In dissemination services, a set of queries is evaluated over a dataset (i.e. a message, a document) in order to identify which *queries* are satisfied by the data.

We present a list of applications where content-based dissemination services may be (or are already) applied. This list of applications is not limited.

**Insurance industry.** Insurance companies are usually composed of different branches that are spread over a region, a country or even the world. The company offices may be linked by an overlay network of content-based routers, which can disseminate new clients (to the employees) and new deals (to the clients) [8].

**Security alerts.** A recent work on automatic containment of worm spread has proposed approaches that function on the network level [3]. The central idea is to analyze network traffic, identify worm features on the network packets, and avoid that such contaminated packets spread to the users. Such scenario may be defined as a “reverse dissemination system” in which the identified messages are *not* delivered to the users or are simply put on quarantine in a specific machine.

**Air traffic control.** Another application is air traffic control [13]. In such services, a traffic control system receives the aircraft situation feed that provides detailed information about the state of airspace. The messages that are disseminated may include information on flight plans, departures, flight location, and landings.

Our work is ideal for disseminating content over all those systems. Since each scenario may have one (or more) specific, distinguished feature, it is possible that some arrangements need to be done. Nevertheless, this set of such diverse scenarios illustrates the potential uses for TRICOT.

## 3. TRICOT Overview

In content-based dissemination systems, queries and data are in constant update. New queries are requested and new data are added frequently. Therefore, the main feature of such a system (from the databases point-of-view) is scalability in terms of number of queries and volume of data that are processed at the same time. In this context, this section introduces TRICOT, a mechanism based on ontologies for disseminating documents, grouping queries, and evaluating queries in XML content-based dissemination services. The main advantage of TRICOT is to provide an homogeneous mechanism for a system of high complexity. Moreover, TRICOT leads the way to integrating dissemination services to other ontology-based services (such as those for integration systems).

In content-based dissemination systems, query evaluation is performed through filtering, in which the queries that a document satisfies are identified. Usually, multi-query processing is performed (a set of queries is evaluated at the same time). Definition 1 presents the message filtering operation that is considered in our work.

**Definition 1** (*Message Filtering*). Let  $D$  be a set of XML documents received through an infinite stream of documents  $d_i$ ,  $i = 1, \dots$ . Let  $Q$  be a set of queries  $q_i$ ,  $i = 1..n$ . The message filtering operation identifies a sub-set of  $Q$ , denominated  $Q'$ , that contains all the queries satisfied by a document  $d_i$ .

The first step in designing a dissemination system is to choose the base architecture for the routers (network units that will filter documents). Choosing the right architecture is important because it will affect the global performance of the system. Given the size of such systems, a centralized architecture is not enough. Hence, a better option is to employ the structure provided by P2P networks [17]. The advantages of such architecture for the system are: the distributed processing reduces network traffic; usually, the depth of the routing path is also usually reduced; it preserves its features in case of failure and changes on the network. Once the architecture is defined, it is necessary to decide how to evaluate the queries over the documents.

The filtering operation is the central procedure in content-based dissemination systems. Therefore, this paper focuses on the aspects of disseminating documents, grouping queries and evaluating them over the documents. The ultimate goal is to improve the performance of the system as a whole. Those three tasks employ TRICOT, whose main operations are summarized as follows.

TRICOT is formed by the following three operations (which add the *Triple* to TRICOT's name):

1. *Document distribution*. Incoming documents are spread over the peers based on the matching between those documents and the ontologies within the super peer.
2. *Query grouping*. Each peer in the network has a set of queries, which will be evaluated over incoming documents. That set of queries is clustered according to the peer's ontology (detailed in the next section).
3. *Message filtering*. Once the peer has its set of queries clustered, it processes each incoming document with the right set of queries (i.e. the queries that belong to the same ontology domain as the documents), according to Definition 1.

Notice that even though different peers may belong to the same domain, those peers can still have different sets

of queries. Moreover, any XML filtering algorithm may be used for processing the queries within each peer. With TRICOT, the distinguished advantages are twofold: the filtering algorithm will have less queries to process over less documents; and it will process documents over queries of the same domain.

## 4. Documents and Queries with Ontologies

In this section, we detail how TRICOT operates for grouping documents and queries according to ontologies.

### 4.1. Ontologies and Documents

Once a document has arrived, TRICOT needs to identify its domain ontology or generates a new one (in case no appropriate ontology exists). To do so, each super peer has an ontology manager for providing information about existent ontologies and shared documents.

There are three main processes on the ontology manager: (i) the file/ontology matcher - identifies the ontology for a document (which also verifies if there is already an ontology for that file); (ii) the ontology generator - for those documents with no associated ontology; and (iii) the query matcher - identifies an ontology for a query. The first two relate to documents and ontologies, and are explained next. The latter relates to queries, and is presented in section 4.2.

**Process 1. Document and Ontology Matching.** The matching task defines the ontology that best describes an XML document, by measuring the overall similarity between both representations. Specifically, given an XML document  $d$  and a set of  $n$  ontologies  $O$  ( $O = o_1, o_2, o_3, \dots, o_n$ ), the matching document-ontology procedure computes the similarity score  $sim(d, O)$ . Then, it chooses an ontology  $o_m$  ( $0 < m \leq n$ ) with the highest score (greater than a threshold  $t$ ) to represent the corresponding file domain application. It is important to notice that in order to evaluate the similarity between files, two types of perspectives are considered: (i) the lexical perspective evaluates the relations between terms by comparing the strings; and (ii) the semantic perspective focuses on the meaning and conceptual correlation among terms. For the similarity analysis, we consider both types, as detailed in [4].

**Process 2. Ontology Generator.** The ontology generator defines the ontology that describes the concepts and relationships in a certain application domain. This process is necessary when an XML document does *not* have an associated ontology. The ontology is created from the schema integration process, as follows. First, we generate the XML schema (XSD file) from the shared file. The XSD schema

is then translated to the OWL format. By applying several integration rules, an integrated schema is created, also represented in OWL format. The complete process is described and implemented by our tool named Ontogen [5].

## 4.2. Ontologies and Queries

The main idea of TRICOT is to process queries of a domain over documents of the same domain. Hence, besides matching documents and ontologies, it also needs to match queries to ontologies. This way, the queries are grouped based on one specific domain in its respective peers.

For matching queries and ontologies it is important to notice that one XML query is a path expression composed of structural constraints. Those constraints include XML element (attribute) names and their relationships (parent/child and ancestor//descendant<sup>1</sup>). Furthermore, this path expression may define constraints on values or predicates, i.e., constraints on the values of the XML elements (attributes).

Structural constraints of an XML query (using XPath for example) may be represented by a query tree, which forms a *partial* representation of an XML document. Also, the relationships ancestor//descendant are considered a special case of parent/child (i.e. the requirement that the child be at the parent's next level is relaxed, making any element at any lower level acceptable). In this case, the matching between query and ontology may be considered as a *special, reduced* case of the matching between an XML document and an ontology. Specifically, given an XML query  $q$  and an ontology  $o$ , this matching returns the percentage of similarity between  $q$  and  $o$  (where ancestor//descendant is considered a special case of parent/child relationship).

With TRICOT, each query is received by the super peer that performs the matching against its set of ontologies. Once the query domain is defined, the super peer forwards the query to all peers of same domain. For example, consider a super peer  $SP$  connected to peers  $P1$ ,  $P2$ , and  $P3$ . Each peer has its own set of domains:  $P1$  has domains  $d1$  and  $d4$ ;  $P2$  has domains  $d2$ ,  $d4$ , and  $d5$ ; and  $P3$  has domains  $d2$  and  $d6$ . Given that  $SP$  matches query  $q$  to domain  $d2$ . Then, it will forward  $q$  to all peers that have other queries from the same domain, i.e., peers  $P2$  and  $P3$ .

With such query grouping process, it is guaranteed that the peer contains only the sets of queries that refer to its documents domains. Therefore, the number of queries to be evaluated is considerably reduced. Instead of the peer evaluate its whole set of queries over each document, now, it can process only the sub-set of queries that belong to the same domain as is documents. Any XML query evaluation algorithm may be employed.

<sup>1</sup>Other relationships, such as next-sibling, may be reduced to combinations of parent/child and ancestor/descendant (following the specification of XPath, <http://www.w3.org/TR/xpath20>)

## 5. TRICOT in Practice

We have implemented the basic algorithms for TRICOT and performed some experimental evaluations in order to show that TRICOT is a viable mechanism. Here, we analyze the quality of the matching between ontologies and XML documents. The implementation is based on a content-based dissemination system simulator. This system simulates the super peer and peers behaviors. We do not consider any network aspect, such as availability and transmission time, since they are not the focus of our work.

### Quality of the Matching Ontology/XML Document.

In order to evaluate how the documents are distributed over the network, we prepared a case study on the quality of the matching between ontologies and XML documents. The input parameters are documents XML, an ontology (defined in OWL) and a minimum similarity threshold (70% - note that, according to some previous experiments [5], we noticed that such number guarantees good precision and recall). The output is the similarity level between two documents<sup>2</sup>.

The ontologies employed define distinct domains, including touristic packages, curriculum and wines. This study considered a total of 10 thousand XML documents. Those documents belong to three datasets: one thousand documents were generated from different taxonomies to describe tourism activities, called *Tour* dataset; one hundred documents describe curricula in the Brazilian Lattes plataforma<sup>3</sup>, called *CV* dataset; the other documents were randomly created from bibliography catalogs, called *BIB* dataset.

We use the similarities measures defined by OntoGen in order to simulated how a super peer will distribute its documents. Therefore, the higher the similarity rate, the higher the probability of the super peer correctly identifying the domain for a document. The results showed a level of similarity of 96.14% to the one thousand documents on tourism and its ontology. In other words, all documents in the *Tour* dataset are lexically similar. The other XML documents (100 from *CV* and 8900 from *BIB*), presented similarity practically null in relation to the touristic ontology – which was very expected.

In order to verify the contrary situation (i.e. when the system does not find an ontology to the documents), we considered the curriculum ontology. The result was no similarity to all documents. That is justified because the Lattes curriculum employs terms that are completely different from the industry curriculum taxonomies. Finally, no document matched the wine ontology, as expected.

<sup>2</sup>The models for ontologies were obtained in [http://protegewiki.stanford.edu/index.php/Protege\\_Ontology\\_Library](http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library), may 2008.

<sup>3</sup>Plataforma Lattes, CNPq: <http://lattes.cnpq.br/>



**Discussion** To the dissemination systems, this result of identifying 96.14% of the documents correctly, means that those documents will be forwarded to the correct peers (those that contain the queries of the same domain). The remaining 4% presented a threshold less than 70%, which indicates that new ontologies need to be defined. The results for matching ontologies and queries were similar. The only difference is that queries are represented as small XML documents. As future work, we intend to study how to get an ideal threshold.

## 6. Related Work

Recent research on XML content-based dissemination has investigated problems related to different parts of the system [6, 8, 11, 14, 16]. The main goal of those works is scalability in relation to the number of queries evaluated, which is achieved by multi-query processing methods [6, 14] as well as early pruning [11]. We refer to [12] for a more recent, complete overview. While the aforementioned works try to improve an aspect or the other, our work considers a global improvement of the system. Specifically, it focuses on aspects of document dissemination, query grouping, and query evaluation on XML content-based dissemination services. These three tasks employ TRICOT, an homogeneous mechanism based on ontologies.

There are techniques based on similarity for matching schemas and ontologies, for example system COMA++<sup>4</sup>. Also, [1] proposes a data model to represent semantic information that matches ontology features to a description, while [15] proposes TIQS, an approach to integrate data that employ semi-automatic matching of schemas based on a pre-defined global conceptual schema. Our concern is to match documents/queries to ontologies as correctly as possible. Given that the documents are written in XML format, we have employed our previously developed tool (called Ontogen) for the matching [5].

## 7. Conclusions

This paper presented TRICOT, an ontology-based mechanism for evaluating queries on content-based XML dissemination over a P2P network. Its main contribution is to provide an homogeneous structure to distribute queries, disseminate documents, and process queries on XML dissemination systems. As future work, we plan to study how to adapt TRICOT to the unique requirements of the mentioned applications, such as the insurance industry.

<sup>4</sup><http://dbs.uni-leipzig.de/Research/coma.html>

## References

- [1] J. Broekstra, M. Ehrig, and P. Haase. A metadata model for semantics-based peer-to-peer systems. In *Work. on Semantics in Peer-to-Peer and Grid Computing.*, 2003.
- [2] X. Chen, Y. Chen, and F. Rao. An Efficient Spatial Publish/Subscribe System for Intelligent Location-based Services. In *Procs. of DEBS*, pages 1–6, 2003.
- [3] M. Costa et al. Vigilante: End-to-End Containment of Internet Worms. In *SOSP*, 2005.
- [4] D. de Brum Saccol et. al. An Ontology-based Approach for Semantic Interoperability in P2P Systems. In *ICEIS*, 2008.
- [5] D. de Brum Saccol et. al. Managing Application Domains in P2P Systems. In *IRI*, pages 451–456, 2008.
- [6] Y. Diao, S. Rizvi, and M. J. Franklin. Towards an Internet-Scale XML Dissemination Service. In *VLDB*, pages 612–623, 2004.
- [7] W. Fenner et. al. XTreeNet: Scalable Overlay Networks for XML Content Dissemination and Querying. In *WCW*, pages 41–46, 2005.
- [8] G. Li, S. Hou, and H.-A. Jacobsen. Routing of XML and XPath Queries in Data Dissemination Networks. In *ICDE*, pages 1400–1404, 2007.
- [9] H. Liu, V. Ramasubramanian, and b. . I. y. . . p. . . Emin Gün, title = Client Behavior and Feed Characteristics of RSS, a Publish-Subscribe System for Web Micronews.
- [10] C. Meghini and N. Spyrtos. Computing Intensions of Digital Library Collections. In *ICFCA*, pages 66–81, 2007.
- [11] M. M. Moro, P. Bakalov, and V. J. Tsotras. Early Profile Pruning on XML-aware Publish-Subscribe Systems. In *VLDB*, pages 866–877, 2007.
- [12] M. M. Moro, Z. Vagena, and V. J. Tsotras. *Open and Novel Issues in XML Database Applications: Future Directions and Advanced Technologies*, chapter Recent Advances and Challenges in XML Document Routing, pages 136–150. IGI Global, 2009.
- [13] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-Based Content Routing using XML. In *SOSP*, pages 160–173, 2001.
- [14] Z. Vagena, M. M. Moro, and V. J. Tsotras. RoXSum: Leveraging Data Aggregation and Batch Processing for XML Routing. In *ICDE*, pages 1466–1470, 2007.
- [15] L. Xu and D. W. Embley. A composite approach to automating direct and indirect schema mappings. *Inf. Syst.*, 31(8):697–732, 2006.
- [16] Z. Xu, N. Seliya, and W. Wu. An adaptive neural network with dynamic structure for software defect prediction. In *SEKE*, pages 79–84, 2008.
- [17] Y. Zhu and Y. Hu. Ferry: A P2P-Based Architecture for Content-Based Publish/Subscribe Services. *IEEE Trans. Parallel Distrib. Syst.*, 18(5):672–685, 2007.

# An Ontology-Based Approach to Portable Embedded System Development

Feng Chen, Hong Zhou, Jianzhi Li, Ruimin Liu and Hongji Yang

*Software Technology Research Laboratory*

*De Montfort University, Leicester, UK*

*fengchen, hongzhou, jianzhli, rliu, hyang @dmu.ac.uk*

Han Li, He Guo and Yuxin Wang

*Dalian University of Technology, Dalian, China*

*han.li, guohe, wyx @dlut.edu.cn*

## Abstract

*There is no doubt that portability of software application is becoming crucial in industrial distributed and embedded software environment. With respect to the ever increasing requirement of system integration, software should be implemented on as many different platforms as possible. Following the basic principles of knowledge-based perspectives, this paper proposes an ontology-based portable software development approach. Firstly, portability technique is discussed. Secondly, the framework and implementation of an ontology-based VRTOS is presented. Thirdly, a selected case study has been conducted, which helps evaluate the proposed approach. Finally, conclusion is drawn and further research directions are advocated.*

**Keywords:** *Ontology, Software Portability, Virtual Real Time Operating System (VRTOS), Embedded System, Portable Operating System Interface (POSIX)*

## 1. Introduction

With the ever increasing complexity of embedded systems, it is desirable to employ Real-Time Operating System (RTOS) to fulfil the requirements of stringent timing and resource constraints of different real-time applications. Even though the embedded system development is supported by RTOS, the general development environment is In-Circuit Emulator (ICE) that has some disadvantages for software development:

- (1) Software and hardware development can not be paralleled.
- (2) ICE is very expensive and does not support multi-users.
- (3) Hardware errors and software errors can not be separated easily during the development.

(4) ICE is a proprietary system without full-featured testing and debugging tool support.

To meet the needs of developing embedded systems, it is an advantage that the software can be developed on general platform, like Windows platform, and be ported to the specific RTOS after the development. Software portability is hence one of the most important issues during embedded system development.

The advantages of portable software have been recognised for many years. As an inherently knowledge intensive activity, software development requires a great number of knowledge, covers from expertise to experience in the application domains [26]. In general, it is expected to reduce the development and maintenance costs and delays through a relative general domain-specific pattern or architecture, which will push the development to goals in adaptability, reusability, and line-product [3, 17]. Since ontology can provide a vocabulary of terms and relations to model such domains, it will therefore facilitate the construction of the domain-specific solutions by introducing ontology-based approach. This paper follows the basic principles of knowledge-based perspectives and proposes an ontology-based portable software development approach, focusing on development of Virtual Real Time Operating System (VRTOS).

## 2. Software Portability

Portability of applications across different platforms is a subject that has attracted a lot of attention for some time. Software portability refers to how easy a software program can be moved between different environment/platform. H. Kaindl describes portability of software by two definitions [13]:

*Definition 1: An Environment  $E$  is a triple  $(\{l_1, \dots, l_i\}, o, m)$ , where  $\{l_1, \dots, l_i\}$ ,  $i \in N$ , is a set of language processors,  $o$  is an operating system and  $m$  is a machine.*

*Definition 2: A port is a mapping  $(P, E(\{l_1, \dots, l_i\}, o, m)) \rightarrow (P', E'(\{l'_1, \dots, l'_i\}, o', m'))$ , where  $P$  is the program*

to be ported,  $P'$  the resulting program after the port,  $E$  the original and  $E'$  the target environment. The following condition must hold:  $(\{l_1, \dots, l_i\} \neq \{l'_1, \dots, l'_i\}) \vee (o \neq o') \vee (m \neq m')$ .

J. Mooney [15] uses the following as a working definition for the attribute of portability:

*A software unit is portable (exhibits portability) across a class of environments to the degree that the cost to transport and adapt it to a new environment in the class is less than the cost of redevelopment.*

Portability does not mean the same thing as porting. Porting involves simply making an existing application run successfully on a new platform and can often result in replacing one set of system dependencies with another. The term portable implies that the software was intended for several platforms from the beginning and that this factor was considered throughout the design and implementation [1].

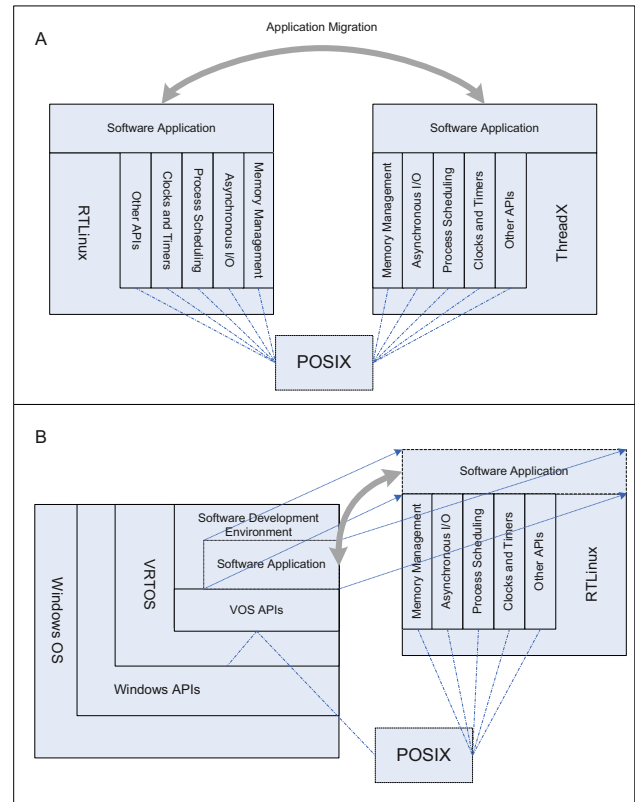
A very important aspect of software application related to portability is its standardisation. Open standards support software portability by providing high level programming models and abstractions. Two levels of standardisation are applied to ensure software portability. One is the standardisation of high-level languages, which are supported across different systems. Another is to use a standard set of application programming interfaces (API) and services which are available on all the target platforms, for example, POSIX interface is itself designed to be portable and POSIX standard contains most of the standard UNIX compatible system call interface.

### 3. Virtual Real Time Operating System

There are a wide variety of options available for the development of portable software applications. The essence of such development has always been related to standardisation, normally implemented by abstraction and isolation. In this research, a Virtual Real Time Operating System (VRTOS) affecting the portability of embedded software application development across different platforms were investigated. A VRTOS here refers to a layer of interfaces and services that resides between the application and the operating system to facilitate the development, deployment and management of embedded system.

The VRTOS is able to handle a wide range of profiles supported in different RTOSs. From the developer's viewpoint, VRTOS provides one RTOS programming environment on another operating system. VRTOS utilises whatever services offered by the underlying operating system to provide uniform services to embedded system. Different implementations of the VRTOS provide the same service interfaces so that software applications become independent from the operating system. Hence, software applications based on VRTOS are portable to

different operating systems environments on the same or different hardware.



**Figure 1. Software Portability and VRTOS**

Figure 1 demonstrates basic scenarios of the VRTOS that supports portability.

In Figure 1A, software application is ported from one RTOS to another one, translation of the system APIs will be the crucial part of this porting process. Different RTOSs provide different APIs. Developers can transform software application based on their knowledge of different platform APIs.

In Figure 1B, Virtual Real-Time Operating System (VRTOS) plays a role as a middleware which is running on Windows. VRTOS supports different RTOS applications so that application software developed on VRTOS can be ported to target RTOS platform directly without any change.

### 4. RTOS Ontology

It is desirable to investigate portability and reusability of platform specific software, during which a great deal of knowledge covering different platforms will be required. To manage such a large number of knowledge, and to utilise it to provide portable software, an ontology-based approach will therefore be introduced that can relieve the burden of application developers from collecting,

classifying and processing information across different platforms.

Ontology defines the kinds of things that exist in the application domain [19]. Providing a shared and common understanding of some domain that can be communicated between people and application systems [6]. In this ontology-based approach, ontology will play a role as RTOS domain knowledge base. Consequently, gathering the knowledge about RTOS from domain experts, and then building the ontology according to this knowledge will be the basic part of the proposed approach.

The RTOS ontology plays the core role for the development of portable software applications with the following reasons:

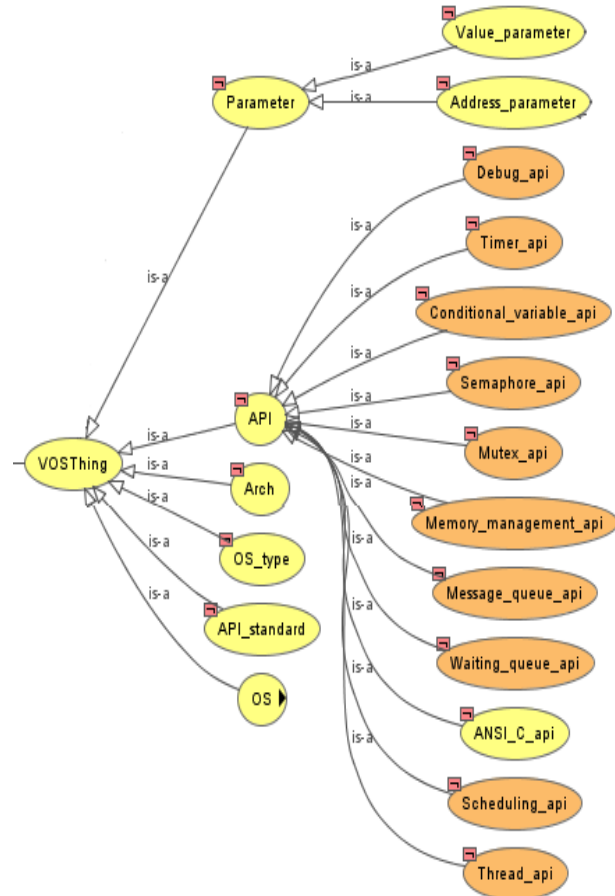
- RTOS ontology provides semantic meaning of the RTOS functions and properties.
- RTOS ontology enables knowledge sharing and further knowledge analysis of different platform.
- RTOS ontology defines a set of common system services which will be used as a standard for portable software development.
- RTOS ontology provides guidance for software porting via knowledge acquisition.

Ontology design is considered as one of the most difficult parts in the proposed approach due to the lack of standard methodologies in ontological engineering domain. A prototype of the operating system ontology was presented in previous work OPTIMA [25, 26], where eight design principles for operating system ontology development has been concluded (Table 1).

Principle	Name
Principle 1	Atomic Concept Definition
Principle 2	Application Concept Recognition
Principle 3	API Based Classification
Principle 4	Service Based Categorisation
Principle 5	Cardinality Restricted Relations
Principle 6	Understanding Aimed Naming
Principle 7	Requirement-Oriented Refactoring
Principle 8	Multi-Layered Structure.

**Table 1. Principles for RTOS Ontology Development**

The RTOS ontology is designed with Protege [20] and stored in RTOS ontological repository. Ontological repository provides ontological knowledge representation as well as small and flexible pieces of code that can be adapted and used to build the RTOS. Feature location technique [2] can be used to build the links of ontology concepts and related source code. Figure 2 demonstrates part of a conceptual structure of RTOS ontology, whose design follows the eight proposed principles.



**Figure 2. Conceptual Structure of RTOS Ontology**

## 5. Realisation of VRTOS on Windows Platform

As mentioned in previous sections, to meet the needs of developing embedded systems, it is desirable that software could be implemented on a general platform and be ported to the specific RTOS after the development. In this Section, how to utilise RTOS ontology to guide the development of VRTOS on Windows platform is discussed.

### 5.1. VRTOS Definition

When the VRTOS is being defined, system analysis will be performed based on knowledge acquisition. With the knowledge retrieval provide by RTOS ontology, the concepts, design policies and mechanisms of RTOSs will be extracted. Such information will be used as guidance for defining programming interfaces of VRTOS.

In Figure 3, each system service could be found in RTOS ontology as an instance. Windows NT system provides system service to create a thread, the API for this service is `CreatThread()`, it is a NON POSIX, the return

type is HANDLE, the parameter for this API is a pointer for the new thread. Furthermore, ThreadX system provide the creating thread service as well, it is invoked by thread\_create(), it is a NON POSIX, the return type is unsigned int, the parameter for it is also a pointer for the new thread.

```

Thread_api(?x) ∧
Thread_service(Thread_service_create) ∧
Windows(Windows_windowsNT) ∧ definedInOS(?x,
Windows_windowsNT) ∧ provideService(?x,
Thread_service_create) ∧ API_standard(?y) ∧
Data_type(?z) → hasAPI(Windows_windowsNT, ?x) ∧
hasAPIStandard(?x, ?y) ∧ hasReturnType(?x, ?z)

Thread_api(?x) ∧
Thread_service(Thread_service_create) ∧
Windows(ThreadX) ∧ definedInOS(?x, ThreadX) ∧
provideService(?x, Thread_service_create) ∧
API_standard(?y) ∧ Data_type(?z) →
hasAPI(ThreadX, ?x) ∧ hasAPIStandard(?x, ?y) ∧
hasReturnType(?x, ?z)

```

Figure 3. Rules for Retrieval of System Service

By system analysis, the VRTOS needs to provide following standard virtual system service and features:

- VRTOS provides application layer a set of uniform system services to perform threading and scheduling, aiming to make the differences among the different target operating systems become transparent for the developers. VRTOS also provide different scheduling policies and priorities.
- Memory management is one of the crucial features for the application layer. Currently, memory allocation and free are available, memory usage tracking is provided as well.
- Message queue, mutex and semaphore services are developed as system independent part, which is not related to system API on target platform. In addition, timer service is also provided for application layer.

POSIX could be the substructure for the standard services that VRTOS provides. In order to implement VRTOS on Windows Platform, Windows APIs on related POSIX standard are examined. In this scenario, all the APIs in both Windows platform and POSIX standard are queried with proposed RTOS ontology.

### 5.2. VRTOS Implementation

The VRTOS is implemented on Windows 2000 to provide standard virtual system service to manage system resources such as memory, thread, mutex, semaphore, message queue, timer etc., and to provide debug and

exception handler as well. The architecture of the VRTOS on Windows platform is shown in Figure 4.

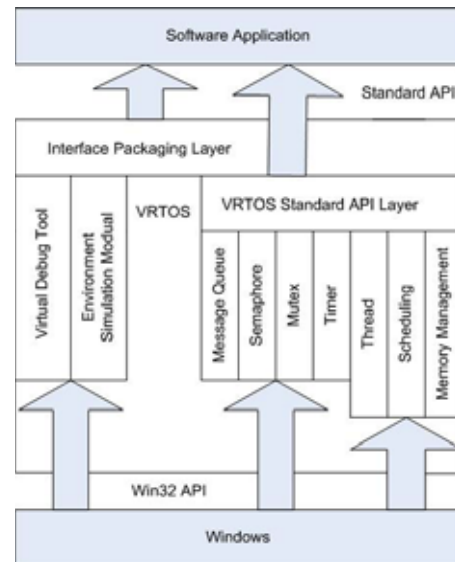


Figure 4. Architecture of a VRTOS on Windows Platform [23]

VRTOS provides a Kernel API Layer, which supports the real-time POSIX Standard [10]. An Interface Layer is designed that can be extended for different RTOSs, such as ThreadX [5] or RTLinux [18]. VRTOS supports the pre-emptive schedule policy of the First-Come-First-Served (FCFS) style and simulates many kinds of system resources. A visual debug tool that enables external environment simulation facilitates the debugging of embedded software greatly [23].

In order to develop VRTOS fast, an open source project, RTLinux, is chosen as the main reference since RTLinux is fully POSIX-compatible. RTLinux is analysed to construct the ontology repository so that the source code of RTLinux can be reused. Meanwhile, the windows APIs on related POSIX standard are examined. All the APIs in both Windows platform and POSIX standard are queried with proposed RTOS ontology. If some POSIX features are not supported by Windows, it means corresponding development is required.

### 5.3. VRTOS Testing

In current VRTOS version, there are 3 system simulation functions, 30 thread functions (10 of them are mutex related function), 2 memory management functions, 7 message queue functions, 6 semaphore related functions and 9 timer related functions. Due to the classification of the system APIs, five sets of 30 testing cases are designed for the VRTOS testing, (i.e. message queue, memory management, mutex, semaphore and timer). Each case includes using thread and scheduling. The testing result shows that the test cases with VRTOS are running

properly on Windows platform and the VRTOS design and development are successful.

## 6. Development of Portable Communication Protocol Stack

To evaluate the proposed approach, the VRTOS has been used for the development of a portable WCDMA 3GPP Protocol Stack, which has been supplied to a number of international mobile phone manufacturers.

WCDMA 3GPP Protocol Stack was designed for several common used RTOSs and has been ported on ThreadX and RTLinux. The whole system was developed by 50 persons for one year. All the development tasks were debugged and tested on VRTOS platform, and only in the final stage, ICE was used for the physical board testing. The experiment result shows that:

- (1) Software can be ported to different platforms directly without any migration task.
- (2) Software can be developed before the hardware development.
- (3) Multi-developers can work in a parallel way.
- (4) Software errors can be separated easily during the development.
- (5) Full-featured testing and debugging tool can be used during the software development.

## 7. Related Work

Portability of software application has been studied for decades of years. In [1, 13, 21], software portability research has been proposed from many aspects, such as program, data, user interface (UI) and documentation, etc. Many factors which hinder software portability have been indicated ranging from hardware platforms to operating system platforms.

Janka [12] presents a new development framework PeakWare for RACE (PW4R), which provides the ability to manage software and hardware libraries which supports software reuse and portability. Vuletic et al. [22] propose a transparent, portable and hardware agnostic programming paradigm to achieve portability and uniform programming by reconfigurable computing. Mosbeck et al. [16] describe software portability in open architectures. They argue that standardised interfaces and a set of common services must be provided to facilitate application portability in open architectures, which are known as abstraction and isolation methods. All of these three researches are similar that they all abstract a set of standard services and create a virtual layer to provide such standard services, leading to the improvement of software portability. However, none of these works indicates or utilise the knowledge intensive features to improve software portability.

Ontology based software development focuses on how ontology can be used to facilitate system analysis, design, coding, implementation, verification and documentation as well. Zimmer and Rauschmayer [27] present a way of enhanced ontology-based software modelling. Their tool TUNA aims to combine XP and MDA by giving MDA rich means for integrating modelling concepts with the source code. Furtado et al. [7] propose a universal user interface design approach which is separated into three levels of abstraction. The creation of the domain ontology is the conceptual level, the elaboration of models is the logical level, and the code transformation is the physical level. Ontology has also been used for program comprehension. Yang et al. [24] suggests that ontology have a great potential for legacy software understanding and re-engineering.

Many researches have been carried out on portability of software applications based on modelling and standardisation of Operating System. Gauthier et al. [8] propose a methodology for automatic generation of application specific operating systems and automatic targeting of application code. Gerstlauer et al. [9] present a RTOS model built on top of existing system level design languages which, by providing the key features typically available in any RTOS, allows the designer to model the dynamic behaviour of multi-tasking systems at higher abstraction levels to be incorporated into existing design flows. Madsen et al. [14] present a modelling framework consisting of basic RTOS service models including scheduling, synchronization and resource allocation, and task model that is able to model periodic and aperiodic tasks as well as task properties. Many world-famous software research institutes are doing research to facilitate porting between Windows and UNIX-like operating systems, e.g., Microsoft Interoperability and Migration Centre, AT&T Labs Research, Cygwin [4], Interix [11], etc.

## 8. Conclusion and Future Work

This paper proposes an ontology-based approach to developing a VRTOS to enhance the portability of embedded system applications across different RTOSs, which fits in the knowledge-based engineering contexts.

Through the RTOS ontology and knowledge representation techniques, the functional equivalence of different operating systems has been established by defining a set of common system services and implementing as a VRTOS. VRTOS has successfully disentangled computing environments from their underlying operating system. Hence, the underlying operating system becomes totally transparent to the software applications, which therefore improves the portability of software applications.

Several advantages have been shown by building and

utilising OS ontology in this paper. However, there are two types of extra costs incurred when using this ontology-based approach to developing portable software applications:

- The costs of developing RTOS ontology. This process is undoubtedly time-consuming endeavour, since a large amount of domain knowledge will be analysed and represented by ontology. However, RTOS ontology is reusable and expandable.
- The costs of implementing VRTOS. Although the effort to develop a VRTOS is large when compared to the effort required when migrating a single program, it is incurred once for different target operating systems within the application domain.

Currently, the RTOS ontology mainly focuses on the programming interface. In the future, this ontology should be expanded from different aspects, and furthermore, it should be published in interactive Semantic Web Community, which enable more experts and specialists working together to contribute to this ontology development.

## References

- [1] A. Bell, "Software Portability," Tessella Support Services PLC Jul. 1998.
- [2] F. Chen, S. Li, and H. Yang, "Feature Analysis for Service-Oriented Re-engineering," presented at 12th IEEE Asia-Pacific Software Engineering Conference (APSEC'05), Taiwan, Dec. 2005.
- [3] F. Chen, H. Guo, L. Dai, et al., "An Application Framework for Ontology-based Data Mining," *Journal of Dalian University of Technology*, vol. Suppl., 43(S1), China, pp. 143-145, Oct. 2003.
- [4] CygnusSolutions, "Cygwin," <http://cygwin.com/>.
- [5] ExpressLogic, "ThreadX User Guide," <http://www.expresslogic.com>, Express Logic, Inc.
- [6] D. Fensel, *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*: Springer-Verlag, Berlin, 2000.
- [7] E. Furtado, J. J. V. Furtado, W. B. Silva, et al., "An Ontology Based Method for Universal Design of User Interfaces," presented at Workshop on Multiple User Interfaces over the Internet: Engineering and Applications Trends, Lille, France, Sep. 2001.
- [8] L. Gauthier, S. Yoo, and A. A. Jerraya, "Automatic Generation and Targeting of Application Specific Operating Systems and Embedded Systems Software," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20(11), pp. 1293-1301, Nov. 2001.
- [9] A. Gerstlauer, H. Yu, and D. D. Gajski, "RTOS Modeling for System Level Design," presented at Design, Automation and Test in Europe (DATE'03), Messe Munich, Germany, Mar. 2003.
- [10] IEEE, "The Open Group Base Specifications Issue 6. IEEE Std 1003.1-2001," The IEEE and The Open Group, 2001.
- [11] InteropSystems, "Interix," <http://www.interix.com/>.
- [12] R. Janka, "A New Development Framework Based On Efficient Middleware for Real-Time Embedded Heterogeneous Multicomputers," presented at IEEE Conference and Workshop on Engineering of Computer-Based Systems (ECBS '99), Nashville, USA, Mar. 1999.
- [13] H. Kaindl, "Portability of Software," *ACM SIGPLAN Notices*, vol. 23(6), pp. 59 - 68, 1988.
- [14] J. Madsen, K. Virk, and M. Gonzales, "Abstract RTOS Modelling for Multiprocessor System-on-Chip," presented at International Symposium on System-on-chip(SoC'03), Tampere, Finland, Nov. 2003.
- [15] J. D. Mooney, "Technical Reports: Bringing Portability to the Software Process," West Virginia University 1997.
- [16] R. A. Mosbeck, L. C. Reeve, and J. R. Thedens, "Software Portability in Open Architectures," presented at IEEE/AIAA 20th Digital Avionics Systems Conference (DASC'01), Daytona Beach, USA, Oct. 2001.
- [17] D. C. Rine and R. M. Sonnemann, "Investments in Reusable Software: A Study of Software Reuse Investment Success Factors," *Journal of Systems and Software*, vol. 41, pp. 17-32, 1997.
- [18] RTLinux, "RTLinux V3.0 Source Code," <ftp://ftp.rtlinux.com/pub/rtlinux/v3/>.
- [19] J. F. Sowa, *Knowledge Representation*: Brooks/Cole, an imprint of Thomson Learning, 2000.
- [20] Stanford, "Protete," <http://protege.stanford.edu/>.
- [21] M. Tanaka, "A Study of Portability Problems and Evaluation," presented at 8th International Conference on Software Maintenance (ICSM'92), Orlando, USA, Nov. 1992.
- [22] M. Vuletic, L. Pozzi, and P. Ienne, "Programming Transparency and Portable Hardware Interfacing: Towards General-Purpose Reconfigurable Computing," presented at 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04), Galveston, Texas, Sep. 2004.
- [23] Y. Wang, F. Chen, L. Xu, et al., "An Implementation of a Multi-Interface Virtual Real-Time Operating System on Windows Platform," *Journal of Dalian University of Technology*, vol. Suppl., 43(S1), pp. 100-102 (in Chinese), Oct. 2003.
- [24] H. Yang, Z. Cui, and P. O'Brien, "Extracting Ontologies from Legacy Systems for Understanding and Re-Engineering," presented at 23rd Annual International Computer Software and Applications Conference (COMPSAC'99), Phoenix, AZ, Oct. 1999.
- [25] H. Zhou, F. Chen, and H. Yang, "Developing Application Specific Ontology for Program Comprehension by Combining Domain Ontology with Code Ontology," presented at the 8th International Conference on Quality Software Oxford, 2008.
- [26] H. Zhou, J. Kang, F. Chen, et al., "OPTIMA: an Ontology-based Platform-specific software Migration Approach," presented at 7th International Conference on Quality Software (QSIC'07), Portland, Oregon, USA, Oct. 2007.
- [27] C. Zimmer and A. Rauschmayer, "Tuna: Ontology-Based Source Code Navigation and Annotation," presented at Workshop on Ontologies as Software Engineering Artifacts, Vancouver, Canada, 2004.

# An Integrated Ontology Framework for Health Information Exchange

S. Demurjian, R. Saripalle, and S. Berhe

Department of Computer Science & Engineering, University of Connecticut,  
U-2155, 371 Fairfield Road, Storrs, CT, USA

Email: {steve,rishikanth,solomon.berhe}@enr.uconn.edu

## Abstract

As we approach 2010, health information technology (HIT) - electronic medical records (EMRs) by providers and personal health records (PHRs) by patients - has begun to pervade all aspects of health care. Moreover, there is an emergent need for health information exchange (HIE) to link disparate data sources that contain medical information in support of efficient and effective patient care. The ability to support HIE across a multitude of sources (EMRs, PHRs, databases, etc.) without widely accepted standards is a difficult, if not monumental task. Issues to be addressed include: maintaining a master patient index, insuring HIPAA-compliant privacy and security, reconciling ontologies at syntactic and semantic levels, assuring data quality and consistency, etc. In this paper, we compare and contrast frameworks and architectures for health information exchange, and based on our assessment, propose a hybrid ontology-based architecture that provides both syntactic integration and semantic unification.

## 1 Introduction

In the next 5 to 10 years, the adoption and usage of health information technology (HIT) systems will be pervasive including: *electronic medical record (EMR)* to manage health-related information for each patient; *practice management system (PMS)* to handle financial, demographic, and billing information; *electronic prescribing (eRx)* to write and transmit prescriptions; provider sponsored *patient portals (PP)* to access lab results, request refills, make, cancel, and confirm appointments, etc.; and, *personal health records (PHR)* that place the control of health-related information directly into the hands of patients (e.g., Google Health [7], Microsoft HealthVault [18], WebMD PHR [27], etc.). From large research hospitals to small physician practices, across all support services (e.g., laboratories, scans, pharmacists, insurers, etc.), there is an emergent need for health information exchange (HIE) to link multiple data sources for ef-

ficient and effective patient care. In fact, the American Recovery and Reinvestment Act of 2009 includes \$2 billion for the Office of the National Coordinator for HIT [21] and up to \$36 billion to states (5 to 1 federal/state match) focused on HIT adoption including EMRs, PHRs, PMSs, PPs, and HIE. All of these systems must adhere to stringent Health Insurance Portability and Accountability Act (HIPAA) regulations [9] for the security, availability, transmission, and release of a patient's medical information.

The movement to a massively linked health information network, will be accompanied by dramatic changes in the way that health care providers (e.g., physicians, nurse practitioners, nurses, rehab centers, hospitals, therapists, etc.) communicate with one another, and patients and their families [1, 2]. Since patients move from provider (internist) to provider (cardiologist) and from their home to a hospital to a rehab center and back home again, communication and workflow are key issues [8, 11, 25], with HIE facilitating remote access to information distributed among multiple EMRs, PHRs, and PPs. To accomplish HIE, each of these individual systems are likely to maintain their own *ontologies* to capture the meaning of stored information. As a result, HIE must consider the merging of ontologies, from syntactic and semantic perspectives, since the same term in two different ontologies (from two different systems) may be interpreted differently. The intent is to create a consistent, complete, and historically accurate patient medical record, known as a *virtual chart (VC)* [13], shown in Figure 1.

In this paper, we explore issues and approaches for HIE that unifies information into a VC, by addressing both syntactic integration and semantic unification across EMRs, PHRs, PPs, etc., as given in the top portion of Figure 1. Specifically, we propose an ontology-based framework that can serve as a blueprint for HIE, which has been influenced by a comparison and analysis on the work of other researchers [24, 26, 28, 29]. By examining and contrasting these approaches based on various criteria, we are able to understand their deficiencies, and leverage this understanding to propose an ontology-based framework. In the re-



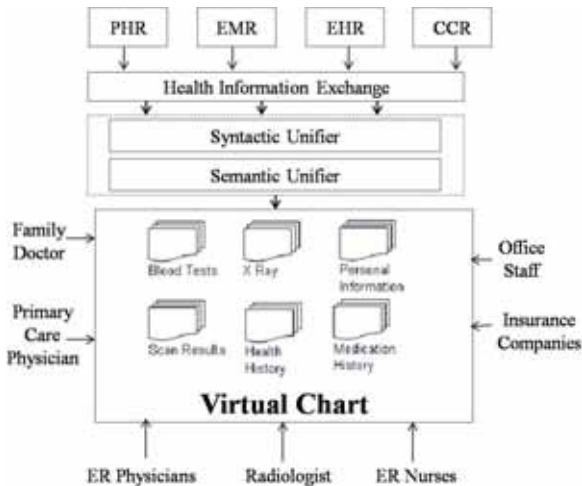


Figure 1: HIE and the Virtual Chart.

remainder of this paper: Section 2 details HIE requirements; Section 3 presents, compares, and contrasts the four alternative architectures against a set of evaluation criteria; Section 4 proposes a hybrid architecture that provides an extensible, integrated ontology framework for HIE; and, Section 5 offering concluding remarks and details planned work.

## 2 HIE Requirements

This section explores various requirements for health information exchange (HIE). As shown in the bottom of Figure 1, stakeholders access required information via a *virtual chart (VC)* [13]. Such information is gathered from different repositories (e.g., a physician’s EMR, a hospital EMR, a patient’s PHR, a pharmacy system, etc.) by HIE. In a futuristic health care scenario, consider Mr. J. Smith, a 78 year old patient with diabetes and a history of smoking who presents to the ER with shortness of breath and wheezing for the first time. Mr. Smith will be treated by individuals who are both in person and located remotely: ER physician, ER nurses, patient’s primary care physician, radiologist, cardiologist, hospitalist (if the patient is admitted, manages day-to-day care), in-hospital pharmacist, insurer (who is in the loop to determine/approve length of stay), discharge planner (to arrange care and give instructions on discharge), external pharmacy (medications after discharge), and visiting nurse (in-home services). Each stakeholder, at different points in time, need access to the VC. The challenge with HIE is to integrate medical data across all of these different systems. Note that all of the previously indicated systems used by the stakeholders are available and in place, often in isolation and with limited HIE.

Second, syntactic unification must be supported for data integration in HIE, in approaches such as: *Global-as-View (GaV)* and *Local-as-View (LaV)* [16]. In LaV, the concepts

in the local schema (e.g., from an EMR) are mapped to a global schema (e.g., for the VC) via queries that operate as mapping rules between the two schemas. The LaV approach favors the extensibility of HIE: adding a new source simply means enriching the mapping with a new assertion, without other changes. In GaV, each concept in the global schema has a corresponding query or mapping rule over the local schema, indicating the method needed by the sources to retrieve data. Adding new sources may have an impact on elements of the global schema requiring redefinition.

Third, in the above scenario, multiple stakeholders communicate on J. Smith and analyze his data to determine the cause and treatment via the VC. Since the data formats of the sources may follow various standards (e.g., HL7 [10], HIPAA [9], ICD9/ICD10 codes [12], etc.), both the structural and semantic data from heterogeneous sources must be unified to represent the data in the VC, as shown in the top of Figure 1. The VC provides customized access based on each stakeholder for both viewing and modifying of this unified data set. To illustrate, consider Figure 2 that has a medical file from a hospital in an HL7 XML format that follows the semantics of the hospital and a second file from a medical center also in an HL7 XML format which instead follows ICD codes. This data doesn’t need syntactical unification, but must be semantically unified; the dashed arrows are semantic equivalences (e.g., A to ICD10, C to ICD11, etc.). Note that documents with different formats (relational table and HL7) must be first syntactical unified.

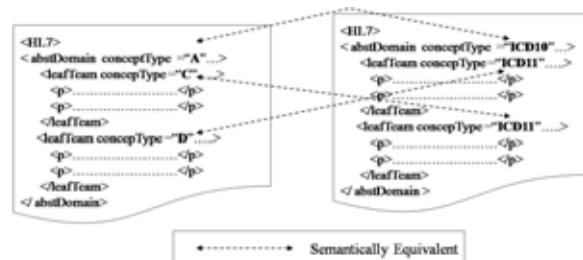


Figure 2: Semantic Unification Example.

Fourth, privacy, security, and consent for the VC must be addressed, including: who has the right to access information, the extent of patient control over their own data, and the way data is stored or linked to other data sources. These decisions must be guided by law [9] and security models such as the National Institute for Standards and Technology role-based access control (RBAC) [19]. HIPAA permits hospitals and medical centers to share protected health information for treatment, payment, and research (quality improvement) with patient authorization; however, it is unclear if the current authorization process is sufficient to support wide-scale regional and/or national HIE.

Finally, workflow, akin to a virtual office setting [3], must be addressed in a medical context, since future pa-

tient care must be handled in a different manner than current practice [14]. Traditional workflow systems lack methods to define relationships among objects in service and the logics of services, to differentiate between the responsibilities and actions providers. Most workflow systems have predefined flows that cannot be dynamically changed. As indicated in [14], medical cases cannot be anticipated; J. Smith's history and symptoms are unique. As a result, workflow for the VC must adapt to changing medical situations, domain knowledge, and medical treatments.

### 3 HIE Candidate Architectures

Health information exchange (HIE) combines data sources in order to provide a unified view to end-users, application components, etc. In this section, we explore, compare, and contrast four candidate architectures [24, 26, 28, 29] to achieve HIE integration via a global schema approach (recall GaV and LaV [16] from Section 2). To assist in this process, we define five evaluation criteria: Syntactic Integration, Semantic Unification, Ontology Support, Adaptability, and Extensibility. *Syntactic Integration* considers the ability of the architecture to unify multi-structural information into a global structural format. *Semantic Unification* represents the ability of the architecture to identify, from heterogeneous data sources, semantically equivalent concepts, properties, etc., which can then be grouped together. *Ontology Support* considers the degree that a given architecture supports semantic interpretations and semantic domain knowledge representation via ontologies; this is important since many of the systems to be integrated utilize ontologies that must be reconciled. *Adaptability*, refers to the ease in which the architecture can handle changing and emerging health care standards and regulations; additions to HL7 [10] or to HIPAA [9] may have significant impact. Finally, *Extensibility* refers to the degree that the architecture can be extended with new data sources and its ability to handle emerging requirements. For the latter, this includes, new standards for HIE, support for collaboration and workflow, new security models, semantic reasoning, etc.

The first alternative architecture [26] in Figure 3 is an XML and XSLT based heterogeneous database integration scheme using a GaV approach. When a user poses a global query, it is decomposed into subqueries to the various sites (bottom of Figure 3) which are processed with data returned in an XML format. XSLT and metadata from each local database are used to transform local results into the global schema format. The middle layer collects the transformed data from all sources and integrates into a consolidated view. The global schema is generated by unifying structurally equivalent XML schema's; this is full support of Syntactic Integration. This architecture uses metadata, and local/global schema information for a limited semantic

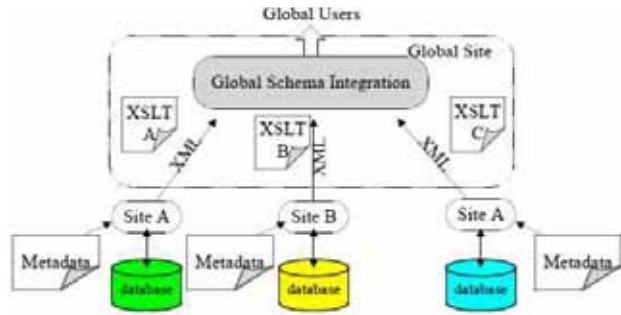


Figure 3: Alternative 1: Tseng.

interpretation of database attributes, but this does not explicitly satisfy Semantic Unification and Ontology Support. Adaptability is supported in a limited sense; the key for data integration is in the XSLT files that specify the mapping at the XML level. These files essentially contain the mapping (e.g., say from HL7 version 1 in Site A to HL7 version 3 in global sphere), and if there are changes, then these XSLTs must be changed. Finally, Extensibility to add new sources requires a new XSLT for each source; however, more complex extensions such as collaboration, semantic reasoning, etc., are not supported.

The second alternative architecture [28] in Figure 4 unifies heterogeneous databases using a semi-automatic technique for semantic integration using schemas, data types, constraints, etc., augmented by a multi-step process of parsing, classification (grouping of similar attributes), and training of the neural network to identify similar attributes. To accomplish these tasks, database concepts and schema are input to a self-organizing mapping algorithm to categorize attributes with the output of classifier used as the training data (to determine similarity of attributes) for a category learning and recognition algorithm. This architecture is very similar to alternative 1, with Syntactic Integration fully supported. However, the assumption is that the database architect provides the meaning of full names and metadata for the databases attributes, thereby supporting only a limited degree of Semantic Unification by grouping similar terms and properties. There is no usage of Ontology Support, and limited Adaptability and Extensibility. The difficulty in this approach, is that in practice, with 10s or even 100s of data sources, the ability to parse, classify, and train becomes computationally prohibitive.

The third alternative architecture [29] in Figure 5 is an ontology-based approach using GaV for integrating XML resources via mappings between the local source schemas and global ontology. The global ontology is expressed in terms of the Resource Description Framework Schema (RDFS) [23], a general purpose language for representing information on the web, including ontologies. In the approach, heterogeneous XML sources are transformed

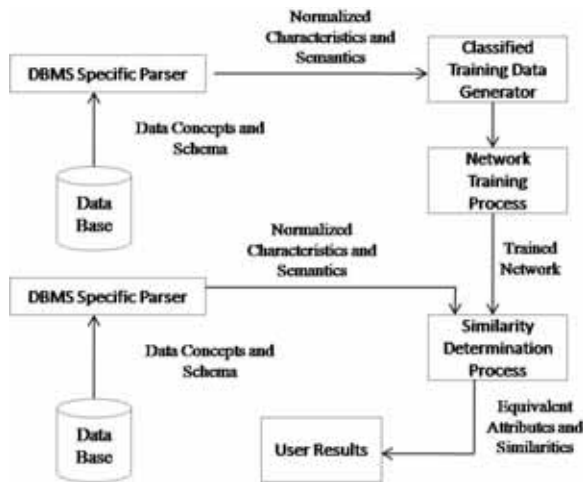


Figure 4: Alternative 2: Li & Clifton.

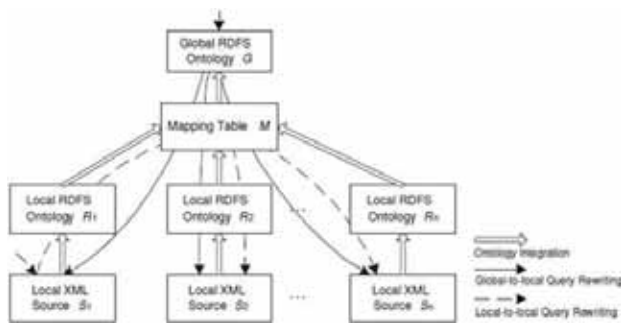


Figure 5: Alternative 3: Huiyong & Cruz.

into local RDFS ontologies, and then merged, via a semi-automatic process, into the global ontology. As discussed in Section 2, generating a global view results in a mapping between concepts in the global ontology and concepts in the local ontologies; this is full Ontology Support. This architecture also fully supports Adaptability, since the ontology mappings deal with both Syntactic Integration and Semantic Unification. As a result, limited Extensibility is supported; including additional data sources is a straightforward process, with more complex extensions problematic.

The final alternative architecture [24] in Figure 6, based on GaV, is the evolvable view environment (EVE), which allow views on heterogeneous data sources to survive schema changes of their underlying data sources while also adapting to changing data in those sources. EVE adapts to the changing data source schema by: *view synchronization* that applies view query rewriting algorithms; and, *view adaptation* in an incremental manner for view definition changes. Unlike other systems, in EVE, query rewriting was relaxed as a means of retaining the validity of a data warehouse in a changing environment. This is achieved via an extension to SQL, that allows a differentiation among

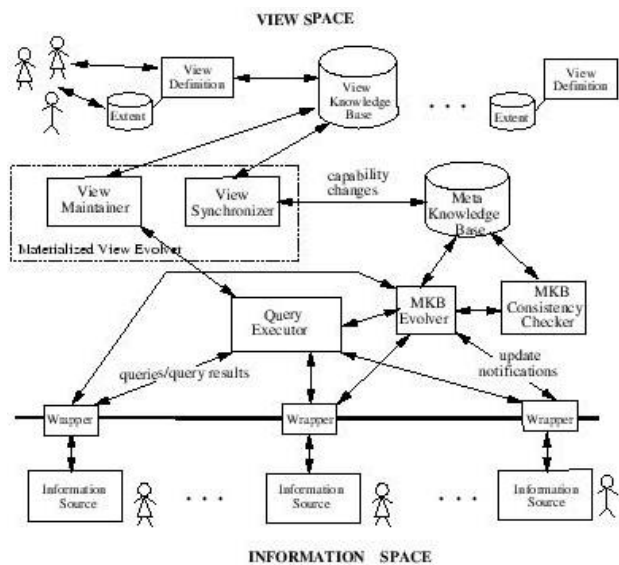


Figure 6: Alternative 4: Rundensteiner, et al.

attributes in a query. Similar to Alternative 3, there is full Syntactic Integration and Semantic Unification, limited Ontology Support, full Adaptability, and limited Extensibility.

To summarize, Table 1 contrasts the architectures against the criteria, where: None means no support, Full means (near) complete support, and Limited (Ltd.) means a degree of support. The final two alternatives are comparable to one another and superior to the first two. The problem

Table 1: Architectural Comparison.

	Alt. 1	Alt. 2	Alt. 3	Alt. 4
Synt. Int.	Full	Full	Full	Full
Seman. Unif.	None	Ltd.	Full	Full
Ontol. Supp.	None	None	Full	Ltd.
Adapt.	Ltd.	Ltd.	Full	Full
Extens.	Ltd.	Ltd.	Ltd.	Ltd.

with all four approaches is that they are incomplete; for HIE, the data sources will include databases, EMRs, PHRs, PPs, etc. Alternatives 3 and 4, while showing promise, are only solving a part of the problem. Generalized information exchange and interoperability transcend database interactions; event-driven architectures, service-oriented architecture, edge-servers for data sharing, etc., must be considered in conjunction with the ideas for the architectures given in this section to arrive at an acceptable solution for HIE.

## 4 Proposed Hybrid Architecture

This section presents a proposed hybrid architecture for health information exchange (HIE) that is centered around

an integrated ontology framework. As shown in Figure 7, the proposed architecture integrates an EMR, PP, and PHR with an *Ontology Engine* via an Event-Driven Architecture (EDA) [4]. An EDA provides mechanisms to completely decouple event sources (EMR, PP, and PHR) from event sinks (Ontology Engine) and these interactions can occur asynchronously. EDA facilitates the inclusion of new sources and supports replication for continued operation during failures. The Ontology Engine comprise one facet of the Virtual Chart (bottom middle), which in turn has a Service Oriented Architecture (SOA) [6] to provide access to VC to the intended user base. SOA complements EDA to provide a set of accessible services for use by developers who are constructing applications for health care providers against the virtual chart (VC).

The specifics of the proposed architecture are as follows. First, each of the data sources (right side of Figure 7) are in control of the information to be published from their database, and provides a *Local Ontology* (terms of the data source) as indicated by  $O_{PHR}$ ,  $O_{EMR}$ , and  $O_{PP}$ . The alternative architectures discussed in Section 3 focused on a global ontology that is obtained by integrating multiple *local ontologies*. However, generating a single global ontology can be tedious and error prone, with a result that is inconsistent, inaccurate, and difficult to browse or edit [17]. The Ontology Engine collects the Local Ontologies (e.g., XML schemas) and processes them through the *Syntactic Unifier* to yield similar structural organization and the *Semantic Unifier* to reconcile the meanings resulting in a *Global Ontology*,  $O_{GO}$ , shown on the left hand side of Figure 7. Similarly, the ontologies associated with each portion of the VC (e.g.,  $O_{BT}$ ,  $O_{HH}$ , etc.) which contain standard medical and supportive ontologies (e.g., UMLS, GALEN, BFO, WordNet, ICD, etc.) are glued together through a set of mapping ontologies (e.g.,  $O_{M1}$ ,  $O_{M2}$ , and  $O_{M3}$ ) to generate a combined ontology  $O_{VC}$ . These two resulting ontologies,  $O_{GO}$  and  $O_{VC}$ , are mapped between one another to form  $O_{GO-VC}$ ; the *Semantic Unifier* is utilized to reconcile meanings as necessary. The resulting ontology,  $O_{GO-VC}$ , contains all of the relationships among the concepts in the local source ontologies (from PHR, EMR, and PP) as well as those from the VC. While these ontologies are similar to the alternatives in Section 3, our approach proposes to store the mapping rules in an ontology to leverage automatic reasoning capabilities.

Finally, the hybrid architecture given in Figure 7 is highly geared towards extensibility. At an architecture level, this is achieved via EDA and SOA. But there are other levels of extensibility. Consider that each of the sources (PHR, EMR, PP, etc.) could be augmented with the inclusion of a Local Security Policy document which allows a data source to set its security requirements. Figure 7 would then be augmented with a Security Engine (akin to the On-

tology Engine), where the local policies are combined into a Global Security Policy that is enforced against the VC. This pair of Local Security Policy and Security Engine is a blueprint for extensibility. For instance, recall in Section 2, that one of the HIE requirements was support for collaboration. Since collaboration in health care is increasing in importance, it will be vital to provide a Local Collaboration Policy for each data source and a Collaboration Engine with a unifier, Global Collaboration Policy, and Workflow Mechanism to interface to the VC.

## 5 Conclusion/Ongoing Work

Health information exchange (HIE) is a complex problem that will require the interaction of researchers and practitioners from multiple disciplines as one seeks to integrate EMRs, PHRs, PPs, databases, and other systems, into a virtual chart for use by health care providers. In this paper, in Section 2, we detailed HIE requirements, including a health care scenario, syntactic/semantic unification, and other issues. Using this as a basis, Section 3 reviewed, compared, and contrasted four architectures based on evaluation criteria. Our analysis led us to propose a hybrid architecture that is ontology-based in Section 4, that includes an Ontology Engine to integrate local ontologies into a Global Ontology.

The work presented herein is an initial step on a long path. A workable solution for HIE will need to consider generalized architectural paradigms. While we used event-driven and service-oriented architectures, we are considering other possibilities (e.g., edge-servers for data sharing, cloud computing, etc.). As a result, our ongoing work is in two directions. First, we are exploring a larger-scale architectural view with an emphasis on identifying other facets of HIE (e.g., security, collaboration, decision support, etc.) that are critical for health care. Second, we are focusing on semantic integration and reconciliation for ontologies, to model dependencies within and across ontologies that may simplify this process.

## References

- [1] J. Abraham and M. Reddy. "Moving Patients Around: A Field Study of Coord. Between Clinical and Non-Clinical Staff in Hospitals." *Proc. of ACM 2008 Conf. on CSCW*.
- [2] R. Agrawal, et al. "Enabling the 21st Century Health Care Information Technology Revolution." *Comm. of the ACM*, 50(2), 2007.
- [3] R. Carbon, et al. "Mobility in the Virtual Office: A Document-Centric Workflow Approach." *Proc. of 1st Intl. Workshop on Software Architectures and Mobility*, 2008.

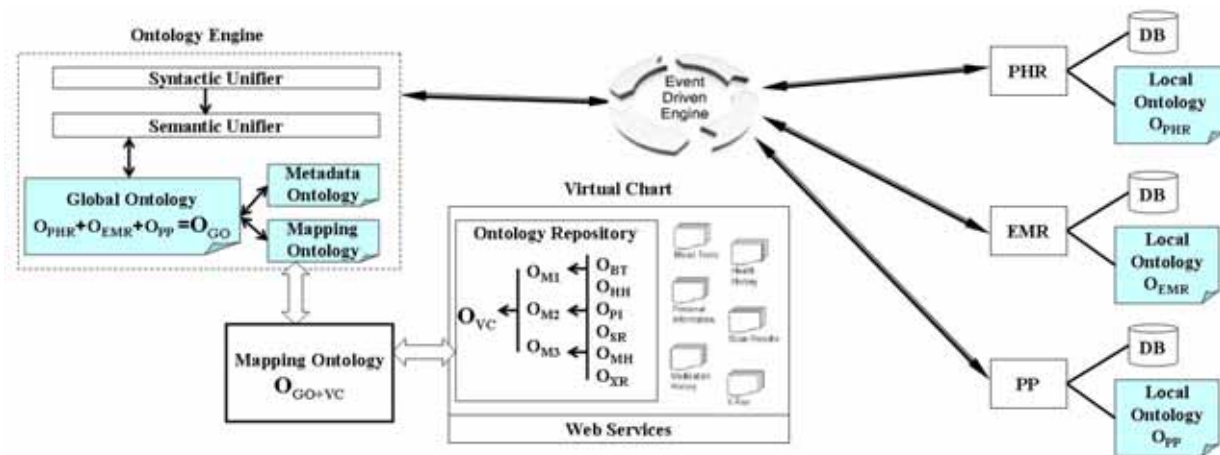


Figure 7: A Hybrid Architecture for Integrating Heterogeneous Sources using Ontologies.

- [4] M. Chandy, et al. "Event Driven Architectures for Distributed Crisis Management." *Proc. 15th IASTED Intl. Conf. on PDCS*, 2003.
- [5] S. Demurjian, et al. "Emerging Trends in Health Care Delivery: Towards Collaborative Security for NIST RBAC," submitted to 23rd IFIP WG11.3 Conference on Data and Applications Security.
- [6] Erl, T. *Service-Oriented Architecture: Concepts, Technology & Design*, Prentice Hall, 2005.
- [7] Google PHR, [www.google.com/intl/en-US/health/tour/index.html](http://www.google.com/intl/en-US/health/tour/index.html)
- [8] L. Hassell and J. Holmes. "Modeling the Workflow of Prescription Writing." *Proc. of 2003 ACM Symp. on Applied Computing*, 2003.
- [9] HIPAA, [www.hhs.gov/ocr/hipaa](http://www.hhs.gov/ocr/hipaa)
- [10] Health Level 7, [www.hl7.org](http://www.hl7.org)
- [11] K. Hoshi and J. Waterworth. "Effective Collaboration for Healthcare by Bridging the Reality Gap across Media-Physical Spaces." *Proc. of 1st Intl. Conf. on Pervasive Technologies Related to Assistive Environments*, 2008.
- [12] ICD 9/10 Codes, [www.cdc.gov/nchs/icd9.htm](http://www.cdc.gov/nchs/icd9.htm)
- [13] P. Kenny, et al. "Virtual Humans for Assisted Health Care." *Proc. of 1st Intl. Conf. on Pervasive Technologies Related to Assistive Environments*, 2008.
- [14] M. Kobayashi, et al. "Work Coordination, Workflow, and Workarounds in a Medical Context." *Ext. Abstrs. on Human Factors in Computing Systems*, 2005.
- [15] A. Koeller. "Integration of Heterogeneous Databases: Discovery of Meta-Information and Maintenance of Schema-Restructuring Views." *Ph.D. Thesis*. UMI Order No.: AAI3030945, 2001.
- [16] M. Lenzerini. "Data Integration: A Theoretical Perspective." *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symp.*, 2002.
- [17] E. Mena, et al. "Observer: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Preexisting Ontologies." *Proc. of First IFCIS Intl. Conf. on Cooperative Information Systems*, 1996.
- [18] Microsoft PHR, [www.healthvault.com](http://www.healthvault.com)
- [19] NIST RBAC, [csrc.nist.gov/rbac/sandhu-ferraiolokuhn-00.pdf](http://csrc.nist.gov/rbac/sandhu-ferraiolokuhn-00.pdf)
- [20] OASIS XACML: [xml.coverpages.org/xacml.html](http://xml.coverpages.org/xacml.html)
- [21] Office of Natl. Coordinator, [www.hhs.gov/healthit](http://www.hhs.gov/healthit)
- [22] P. Patel-Schneider and J. Simeon. "The Yin/Yang web: XML syntax and RDF semantics." *Proc. of the 11th Intl. Conf. on WWW*, 2002.
- [23] W3C RDFS: [www.w3.org/TR/rdf-schema](http://www.w3.org/TR/rdf-schema)
- [24] E. Rundensteiner, et al. "Evolvable View Environment (EVE) : Non-Equivalent View Maintenance under Schema Changes." *Proc. of 1999 ACM SIGMOD Conf.*
- [25] W. Tolone, et al. "Access Control in Collaborative Systems." *ACM Computing Surveys*, 37(1), 2005.
- [26] F. Tseng. "XML-Based Heterogeneous Database Integration For Data Warehouse Creation." *Proc. of Pacific Asia Conf. on Information Systems*, 2005.
- [27] WebMD PHR, [www.webmd.com/phr](http://www.webmd.com/phr)
- [28] Wen-Syan Li, and Chris Clifton "Semantic Integration in Heterogeneous Databases Using Neural Networks." *Proc. of the 20th VLDB Conf.*, 1994.
- [29] H. Xiao, and I. Cruz. "Integrating and Exchanging XML Data using Ontologies." *Proc. of 8th Intl. Database Engineering & Applications Symp.*, 2004.

# Modeling user interpersonal stances in affective dialogues with an ECA

Nicole Novielli, Enrica Gentile

Dipartimento di Informatica, University of Bari  
Via Orabona, 4 – 70125 – Bari (BA) - Italy  
{ novielli, gentile }@di.uniba.it

**Abstract.** Since several forms of anthropomorphic behavior of users towards technologies have been demonstrated, ongoing research on intelligent interfaces aims at developing adapting conversational systems which tailor their behavior according to both cognitive and affective features of users. We present an approach for combining results of our previous experience in detecting and modeling interpersonal stances in advice-giving dialogues with an ECA. We sketch an algorithm for adaptation of the behavior of the agent according to the user social attitude and level of engagement.

## 1. Introduction

Ongoing research on intelligent interfaces aims at developing adaptive conversational systems that can both adjust to individual differences among users and track and adapt to changes in key features of users' language and behavior during conversations. Embodied Conversational Agents (ECA) are one of the forms in which this 'intelligent' kind of interaction promises to be effective [1], if the hypothesis that 'characters contribute to more sociable and user-friendly interfaces' is taken for granted [2].

Our long term goal is to build an ECA which employs social and affective skills to engage users in natural dialogues about advice-giving [3]. In the scenario we investigate, the ECA plays the role of an advice-giver in the domain of healthy eating.

Building an effective dialogue strategy is a complex task in which knowledge of the user characteristics is of primary importance. This knowledge may be acquired by *observing* the user behaviour during the dialogue to build a dynamic, consistent model of her mind. This model can be used for adaptation purposes and should combine both affective and cognitive ingredients [4].

In this paper we present an approach for combining results of our previous experience in detecting and modeling interpersonal stances<sup>1</sup>. In particular, we sketch an algorithm for dialogue adaptation in which we combine: (i) a method for modeling the user social attitude, which integrates consideration of the context with language analysis of individual dialogue turns [23]; (ii) a method for detecting the user engagement, which uses Hidden Markov Models (HMMs) for dialogue pattern modeling [26].

The paper is structured as follows: in Section 2 we describe which attitude aspects could be considered as relevant in advice-giving dialogues adaptation and we provide definition for both, the user social attitude and level of engagement; in Section 3 we provide a brief description of the corpus used for defining the interpersonal stance modeling methods; these methods are then described in details in Section 4; Section 5 provided an example of how we intend to combine knowledge about the user social attitude and level of engagement in dialogue adaptation;

conclusions and directions for future work are provided in Section 6.

## 2. Which attitude aspects?

In the recent years, several projects aimed at classifying the user dialogue turns according to a set of either continuous or discrete 'emotional states' or to the estimated value of some basic components of affective states (such as valence or intensity) [7,8,9]. Other studies aimed at recognizing some personality traits of the users from monologs (e.g. extraversion in emails [10]).

Rather than considering emotions, we look at two aspects of affective interaction, the social attitude and level of engagement, which we assume to be key factors for a successful adaptation of advice-giving dialogues.

### 2.1 Social Attitude

Affective states vary in their degree of stability, ranging from long-standing features (personality traits) to more transient ones (emotions). 'Interpersonal stance' are in a middle of this scale. This general concept was named differently in recent research projects: Paiva [11] and Vaknin<sup>2</sup> talk about 'empathy'; e-learning researchers investigate the concept of 'social presence' [12] which received several definitions, from the general one "the extent to which the communicator is perceived as 'real'" [13] to the more ECA-specific one "the extent to which individuals treat embodied agents as if they were other real human beings" [14].

In referring to the social response of users to ECAs, we will employ the term *social attitude*. To distinguish *warm* from *cold* social attitude, we refer to Andersen and Guerrero's [15] definition of 'interpersonal warmth' as "*the pleasant, contented, intimate feeling that occurs during positive interactions with friends, family, colleagues and romantic partners [and] ... can be conceptualized as... a type of relational experience, and a dimension that underlines many positive experiences*".

Recognizing the social attitude of the user is essential for adapting the agent interaction style and behaviour. Adaptation may be beneficial if the user characteristics are recognized properly (as shown in the examples 1a and 1b) but detrimental in case of misrecognition (as in the example 1c).

Researchers proposed a large variety of markers for social stances. In Section 4, we describe a method for social attitude detection based on a combination of results of linguistic analysis and contextual information.

#### Example 1a

**System:** Hi, my name is Valentina. I'm here to help you to improve your diet.

**User:** What should I do? (*neutral attitude*)

**S:** To start, I'm going to ask you some information about your eating habits. What did you eat for dinner, yesterday?

(*the system reacts with a neutral move as well*)

<sup>1</sup> To Scherer, *interpersonal stance* is "characteristic of an affective style that spontaneously develops or is strategically employed in the interaction with a person or a group of persons, coloring the interpersonal exchange in this situation (e.g. being polite, distant, cold, warm, supportive, contemptuous)": <http://emotion-research.net/deliverables/D3e%20final.pdf>

<sup>2</sup> S. Vaknin: On empathy: <http://samvak.tripod.com/empathy.html>

### Example 1b:

S: Hi, my name is Valentina. I'm here to help you to improve your diet.

U: Hi Valentina, my name is Carlo and I'm happy to interact with you  
(*U shows a warm social attitude by friendly introducing himself*)

S: Hi Carlo, good to hear your interest! I hope you'll find our dialogue useful. I'll, first of all, ask you some information about your eating habits.  
(*S reacts accordingly by adding 'warmth' to his move using some small talk*)

### Example 1c:

S: I think your diet is completely wrong.

U: Oh thanks, Valentina! You're so nice! (*with Ironic intonation*)

S: Thanks! (*By smiling*)

## 2.2 Engagement

To be effective, advice-giving cannot be the same to all interlocutors. The Likelihood Model [5] suggests that in different situations of attention and interest, peripheral or central processing channels should be followed, with more or less affective features. Hence, knowledge of the users is essential to increase their information processing ability and interest, and therefore the effectiveness of advice-giving. In this sense, recognition of engagement is of primary importance for adaptation.

Engagement is a concept to which researchers attach a wide range of related but different meanings [16,17,18], which are meant to be coherent with application domain and adaptation purposes.

In our definition of engagement, we take into account the two main tasks addressed in the advice-giving dialogues: the information provision and the actual advice-giving process. In particular, we consider users to be 'highly engaged' when the system succeeds in involving them in the Advice-Giving process (AG users). A lower level of engagement is experienced by users who are mainly interested in the information-giving task (Information Seeking users or IS). Finally, we classify as 'not engaged' (N) users who don't show any interest in the conversation and give a passive and barely reactive contribution to the interaction, by mainly answering the system's questions, very often with general answers (e.g. 'yes' or 'no').

Distinguishing among these three levels of engagement is relevant for adaptation: IS users might be either helped in their information seeking goal or led by the system to get involved also in the advice giving task; AG users might perceive an increased satisfaction if the agent is believable in playing the role of artificial therapist; N users represent a real challenge for the system: their attitude might be due to a lack of interest in the domain or to their being in the 'precontemplation stage' [19].

Basing on the assumption that the level of involvement affects the dialogue pattern, we describe, in Section 4, a methods which exploits conversational analysis techniques for engagement detection and modeling.

## 3. A WoZ corpus of persuasion dialogues

In defining approaches for the recognition of the affective ingredients of the user's state of mind, we integrated psycholinguistic theories with results of empirical studies on natural data [23,26].

In particular, we observed how people actually behave when experiencing different levels of social attitude and engagement in advice-giving dialogues with an ECA.

We performed a set of Wizard of Oz studies in which the ECA played the role of an artificial therapist in the healthy eating domain [20]. The corpus includes overall 60 dialogues

(overall 1700 adjacency pairs of system-user moves) with graduated students in the 21-28 age range. The study was a 2 (written vs. spoken input) by 2 (background in humanities vs in computer science) between-subjects design, balanced for gender.

## 4. Modeling interpersonal stances

Defining a method for affect recognition and modeling requires dealing with several issues: the characteristics of the phenomenon being analyzed, the adaptation purposes, the intended use of the inferred knowledge about the user, the available source of information (also accordingly to the media used for the interaction). Our approach, built in previous research [23,26], is based on a combination of empirical analysis of the corpus of natural dialogues, described in Section 3, and theoretical background about how interpersonal stances affect the interaction.

### 4.1 A dynamic model of the user social attitude

Social attitude is a mid-term affective state which smoothly evolves during the interaction: users may establish with the agent an initial attitude which mainly depends on their personality, background and expectations from the agent herself. This attitude may then gradually change, in valence and intensity, during interaction, according to the agent's behaviour (e.g. how 'pleasant' and 'engaging' the agent is). This change cannot be assumed to be monotonic: both 'negative' and 'positive' events may occur during the dialog, causing variations in the user attitude, in opposite directions: overall, the user attitude in a given phase of the dialog will be a function of the user's stable characteristics, of the dialog history, and of what the agent just did.

Changes in this attitude reflects on the interaction style of users, mainly on their verbal and non-verbal behaviour. Hence, to estimate the level of social attitude, at every time of the interaction, our approach combines both (i) linguistic analysis of the user move and (ii) consideration of contextual information, as well as the dialogue history [23].

The envisaged methodology is the Dynamic Belief Networks (DBNs) formalism, which allows us to deal with uncertainty in the relationships among the variables involved in the social attitude evaluation.

**Linguistic analysis of user moves.** Researchers describe a large variety of markers for social attitude: Swan [21] proposes a coding schema for analysing social communication in text-based interaction which employs affective, cohesive and interactive indicators in online teaching; similar indicators have been suggested by Polhemus et al [13] and Andersen e Guerrero [15]: personal address and acknowledgement (using the name of the persons to which one is responding, stating their name, agreeing or disagreeing with them), feeling (using descriptive words about how one feels), paralanguage (features of language which are used outside of formal grammar and syntax), humor, social sharing (sharing of information non related to the discussion), social motivators (offering praise, reinforcement and encouragement), value (set of personal beliefs, attitudes), negative responses (disagreement with another comment), self-disclosure (sharing personal information), sense of intimacy (use of a common jargon), benevolent or polemic attitude towards the system failure, interest to protract or close interaction.

According to these theories and to a preliminary analysis of our corpus, we defined a mark-up language [22] (see Table 1) for the corpus annotation whose categories correspond to the ‘signs’ of social attitude we aim at recognizing. We employed the results of annotation studies [22,23] as a gold-standard in defining the recognition method.

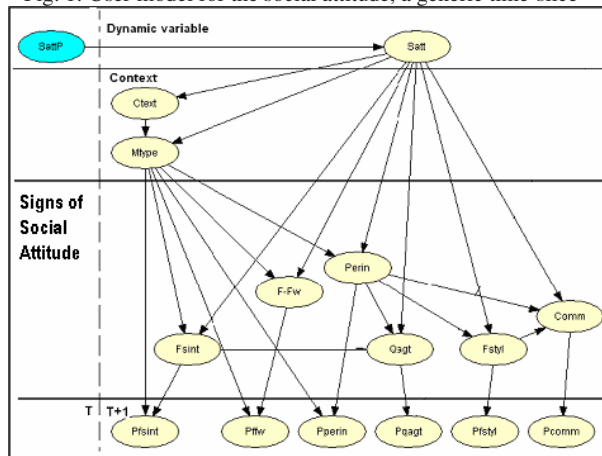
Recognition of linguistic signs of social attitude is performed by using Bayesian classification and can be enriched with acoustic analysis of user move, as described in [22]. A user move is categorized as ‘showing a particular sign of social attitude’ if it contains some word sequences belonging to semantic categories which are defined as ‘salient’ for the considered sign (e.g. expression of greetings or farewell for, respectively, the friendly self introduction and the friendly farewell categories or expression of agreement/disagreement for positive/negative comments).

Tab. 1: Our mark-up language for signs of social attitude

Linguistic cues nodes	DBN node	Signs of social attitude with their definition
Pfsint	Fsint	<b>Friendly self-introduction:</b> The subjects introduce themselves with a friendly attitude (e.g. by giving their name or by explaining the reasons why they are participating in the dialogue).
Pfstyl	Fstyl	<b>Colloquial style:</b> The subject employs a current language, dialectal forms, proverbs etc.
Pperin	Perin	<b>Talks about self:</b> The subjects provide more personal information about themselves than requested by the agent.
Pqagt	Qagt	<b>Personal questions to the agent:</b> The subject tries to know something about the agent's preferences, lifestyle etc., or to give it suggestions in the domain.
Pcomm	Comm	<b>Positive or negative comments:</b> The subjects comment the agent's behavior, experience, domain knowledge, etc.
Pffw	F-fw	<b>Friendly farewell:</b> This may consist in using a friendly farewell form or in asking to carry-on the dialogue.

**Dynamic modeling of Social Attitude.** After every user move is entered, during the interaction, linguistic analysis is performed, producing a set of evidences about linguistic cues of signs of social attitude. The overall value of social attitude is a function of these evidences on signs as well as evidences about context information.

Fig. 1: User model for the social attitude, a generic time-slice



DBNs [24] are local belief networks (called time slices) expanded over time and connected through temporal links. We applied results of the corpus annotation from the annotated data a model of the user social attitude which

includes the dimensions of interest for dialog adaptation [23]. In particular we learnt the temporal part of our DBNs, by considering every single user move in the corpus as an independent observation, using the K2 algorithm [25].

In our DBN (Figure 1) the social attitude is the *hidden* variable, that is the variable we want to monitor (Satt node). It depends on *observable* ones, such as the ‘stable’ characteristics of the users (their background and gender), the context (previous agent move, Ctext, and the user move type, Mtype) and the linguistic cues recognized (leaf nodes of our DBN, first column in Table 1). Intermediate variables are the signs of social attitude listed in Table 1, column two. Links among variables describe the causal relationships among stable characteristics of the users and their behaviour, via intermediate nodes. DBNs, as we employ them, are ‘strictly repetitive models’ in which the Markov property holds, that is the past has no impact on the future given the present. In our simulations, every time slice corresponds to a user move, the stable user characteristics stay unvaried (this is why we omitted the nodes representing the user background and gender from the figure) and temporal links are established only between dynamic subject characteristics in two consecutive time slices.

## 4.2 Modeling engagement with HMMs

The engagement recognition method is based on the assumption that such affective phenomena influence the dialogue dynamics [16]. Hence, we model user categories, by looking at differences in the dialogue pattern [26]. The corpus was labeled so as to classify both system and user moves using categories of communicative acts. These categories are a revision of those proposed in the SWBDL-DAMSL (Switch Board Corpus - Dialogue Act Markup in Several Layers) [27] and are independent from both the task and the domain of the dialogues and only reflect the communicative intention of the move (see Table 3).

Tab. 2: Categories of Wizard and User moves

Tag	Description
<b>Wizard</b>	
OPENING	initial self-introduction by the ECA
QUESTION	question about the user's eating habits
OFFER-GIVE-INFO	generic offer of help or specific information
PERSUASION-SUGGEST	persuasion attempt about dieting
ENCOURAGE	statement enhancing the user motivation
ANSWER	provision of generic information after a user request
TALK-ABOUT-SELF	statement describing own abilities, role, skills
CLOSING	statement of dialogue conclusion
<b>User</b>	
OPENING	initial self-introduction by the user
REQ-INFO	information request
FOLLOW-UP-MORE-DETAILS	further information or justification request
OBJECTION	objection about an ECA assertion/suggestion
SOLICITATION	request of clarification or attention
STAT-ABOUT-SELF	generic assertion or statement about own diet, beliefs, desires and behaviours
STAT-PREFERENCES	assertion about preferences (e.g. food liking)
GENERIC-ANSWER	provision of generic information after an ECA's question or statement
AGREE	acknowledgment or appreciation of ECA's advice
KIND-ATTITUDE-SYSTEM	statement displaying kind attitude towards the system (e.g. jokes, polite sentences, etc.)
CLOSING	statement of dialogue conclusion

To model the differences in the dialogue patterns, we adopt the formalism of Hidden Markov Models (HMMs) [28]. In



our HMM model of a dialogue, the observables are the dialogue act (DA) categories while the path through the hidden states describes interaction evolution. In particular, states represent aggregates of system or user moves, each with a probability to occur in that phase of the dialogue, while transitions represent dialogue sequences, ideally, from a system move to a user move type and vice versa, each with a probability to occur (although in principle, user-user move or system-system move transitions may occur).

The three HMMs representing the three levels of engagement are learnt using, once again, results of an annotation study [26] on the WoZ corpus: two independent raters were asked to annotate the overall attitude of every user by using the labels N, IS and AG. According to the annotation experiment, the corpus was divided into three classes of dialogues, with respect to the label received (N, IS, or AG). Each class is then used for training the corresponding HMM (HMM\_N, HMM\_IS, HMM\_AG), using the Baum-Welch algorithm [29].

When a new dialogue (or dialogue fragment) is entered, it is coded as a sequence of dialogue acts [30]. Then, it is compared with each of the three models to establish which one is more likely to produce such a sequence  $d$  of DAs. The algorithm used for classification is the forward-backward [29], which computes the loglikelihoods: (a)  $\text{Loglik}_N = \log P(d | \text{HMM}_N)$ , (b)  $\text{Loglik}_{IS} = \log P(d | \text{HMM}_{IS})$  and (c)  $\text{Loglik}_{AG} = \log P(d | \text{HMM}_{AG})$ .

The maximum value among (a), (b) and (c) is selected and the case  $d$  is classified as belonging to the corresponding class.

Figure. 2 shows two example HMMs for IS and AG dialogues, respectively. Every dialogue phase is named according to the semantic of the exchanges occurring during the phase itself. 'S' and 'U' are states in which, respectively, the System and the User hold the initiative.

Differences in the probability distributions for both, transitions and dialogue acts observation, describe the differences in the behavior of users belonging to different classes of engagement. In particular, IS and AG users mainly differ in the way they initially approach the dialogue and in the persuasion stage. In the IS model there is an higher probability of directly entering the persuasion phase because of a user request of information. Also, IS users have higher probability of mainly performing a request of information during the persuasion phase, without providing any kind of personal information ('information seeking' phase). On the contrary, in AG models, users are involved in an advice-giving phase. Hence the probability of information requests is lower, in favor of the variety of reactions to system suggestions, according to the users' goal of either enhancing the construction of a shared ground of knowledge about healthy eating, or giving a positive feedback to the ECA. The likelihood of entering the persuasion phase, core of the advice-giving process, after the initial question answering, is high in both models. For a more detailed discussion about modeling attitudes with HMMs, please refer to [26].

The following is a short excerpt from a coded dialogue, in which the systems interact with an IS user.

T(S,1)= Hi, my name is Valentina. I'm here to suggest you how to improve your diet. Do you like eating?

T(U,1)= Hi, my name is Simone and I'm very happy to interact with you.

T(S,2)= Good, let's start then. Do you like eating?

T(U,2) = Yes, I like very much sweets. Are they dangerous?

T(S,3) = Sweets are not particularly healthy and should be limited to special occasion.

T(U,3)= I see.... And what about vegetables?

T(S,4)= Ideally, you should have at least four or even five portions of fruits and vegetables per day.

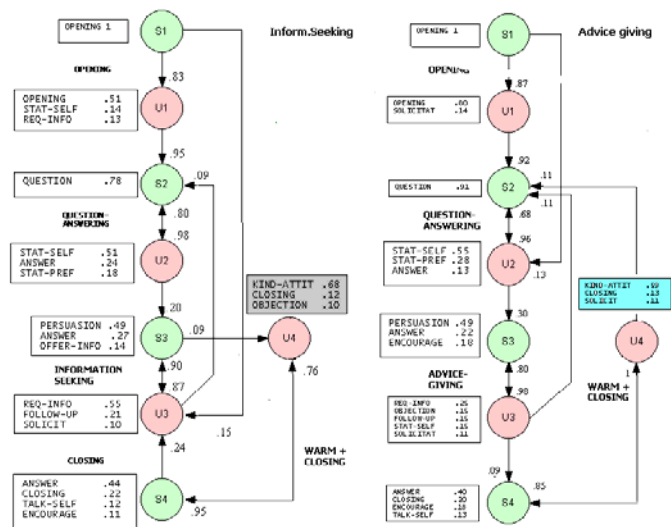
T(U,4)= That's it? What else should I eat to be in a good shape?

The dialogue is coded as follows:  $d = (\text{OPENING}, \text{OPENING}, \text{QUESTION}, \text{REQ-INFO}, \text{PERSUASION-SUGGEST}, \text{REQ-INFO}, \text{PERSUASION-SUGGEST}, \text{REQ-INFO})$ .

The forward-backward computes the values  $\text{Loglik}_N = -90$ ,  $\text{Loglik}_{IS} = -75$ ,  $\text{Loglik}_{AG} = -78$ , hence the dialogue is correctly classified as IS).

According to the IS model in fig. 2, the dialogue  $d$  goes through a very short opening phase to directly enter the information seeking phase, right after the second dialogue turn.

Fig. 2 - HMM models for IS and AG users



## 5. Exploiting interpersonal stances for dialogue adaptation

We sketched an algorithm which describes the dynamic recognition process we intend to implement in our future research. The algorithm is repeated every time a new user move is entered and includes the following steps:

1. A new user move is entered and treated as input for both:
  - (i) a module which implements linguistic analysis to extract linguistic cues for each of the signs of social attitude [22],
  - (ii) a module which classifies the move according to DA taxonomy [30];
2. The attitude model manager sets and propagates linguistic and contextual evidences into the social attitude DBN. The probability of signs of social attitude are evaluated and stored in the agent model of the user state of mind, as well as the overall  $P(\text{Satt})$  value [23]. In the meanwhile, the DA tagging module updates the dialogue history by adding the DAs correspondent to the system and user move at time  $t$ . The HMMs classifier estimates the user level of engagement, using the forward-backward algorithms [26];
3. The agent employs the new knowledge inferred at time  $t$  in order to:
  - (i) plan the content and style of her next move (using the new probabilities for the signs of Social Attitude) and to
  - (ii) revise the overall interaction style and behavior (using the overall value of  $P(\text{Satt})$ ). The new estimation for the level of engagement is also employed to

define/revise the overall dialogue strategy according to the inferred user goal (of either asking for information or being involved in a cooperative process of advice-giving for, respectively, IS and AG users).

The example shown in tab. 3 demonstrates the adaptation algorithm functioning, in particular it shows the two level of analysis we perform at every time slice and how they contribute to the progressive building and updating of the picture of the user state of mind, in terms both of overall value of social attitude and engagement and evidences about

the signs of social attitude detected in the user move and the associated communicative intention.

In the example we simulate the behavior of an empathic agent which reciprocates the user social attitude. We also simulate the condition in which the agent successfully leads an IS user to the AG level of engagement. Though, the correct strategy to employ in adaptation is still an open problem on which we plan to do further investigations in the future research.

Tab. 3: Progressive updating of the system's image of the user mind (Social Attitude and Level of Engagement) during the dialogue

t	Dialogue turn	Signs of Social Attitude (SA)	P(SA)	Dialogue History update	Engagement
1	A: Hi, My name is Valentina: I'm here to suggest you how to improve your diet. U: Hi Valentina: my name is Giulia, and I'm happy to interact with you.	Friendly Self Introduction;	1	S-opening U-opening	(IS)
2	A: Good, I'm glad to hear this from you! Let's start, then. Did you already eat something for breakfast today? U: No	---	.7	S-question U-Generic Answer	N
<i>After a 'neutral' moves, the probability of social attitude decreases, due to the decay simulated by the temporal link in the DBN</i>					
3	A: And for lunch? What do you plan to eat today? U: I'm planning to prepare some meat with rice and vegetables.	---	.55	S-question U- stat about self	N
4	A: What did you have for lunch yesterday? U: I went to a wedding where I ate a lot of fantastic food!!	Talk ab. self, Familiar style	.88	S-question U-stat about self	N
<i>Evidences of linguistic signs of social attitude cause an increase in the overall level of social attitude</i>					
5	A = Ok, now please tell me: is there any particular aspect of dieting you are interested in? I could try to give you some information or suggestion. U: Well... yes... I would like to know what do you mean by 'balanced diet'	---	.6	S-offer-information U-req-info	N
6	A: In a well balanced diet, the 50% of the calories should be taken from carbohydrates. U: I didn't know this, that's interesting Valentina, thanks! And... why are carbohydrates so important?	Pos. comment Familiar style	.9	S-pers-suggest U-follow-up	AG
7	A: Carbohydrates help you to you control your weight. U: Really? I thought carbohydrates were not helpful when trying to loose some weight...	---	.7	S-pers-suggest U-objection	AG
<i>The dialogue goes on until the user asks for information about sweets</i>					
8	A : Sweets are not particularly healthy and should be limited to special occasion. U: But I eat them sometimes, I could not live without sweets, at least once a week!	Friendly Style	.88	S-pers-suggest U- stat about self	AG
9	S: Of course! The pleasure of good eating is important you should not be too rigid in following your diet! U: Good, Valentina! I definitely agree with you!	Familiar Style Pos. Comment	1	S-encourage-sorry U-stat about self	AG

The subject in the dialogue is a female student of humanities, hence the initial probability of observing a warm social attitude is  $P(Satt = Warm) = .61$ .

After the agent self introduction, the user performs a friendly self-introduction in her turn, causing an increase in the overall social attitude. The initial estimation of the engagement is IS, which is the default class, according to our corpus distribution. Due to these evidences, the agent decides to reciprocate the warmth shown by the user, using some small talk in her next move ('Good, I'm glad to hear this from you!') and to start the initial assessment phase of the user situation by asking for information about user's habits and preferences.

The dialogue goes on with the agent reciprocating the user social attitude and the user mainly providing general answers or personal statements, which cause an estimation of a low level of engagement (N). To increase the user engagement, the ECA decides to check her preferences and goals by performing an offer-information move (dialogue turn n. 5). From this point on, the recognized level of engagement

increases (AG), due to the variety of the user behavior (information requests, statement about self and objection) which indicates an high involvement in the advice-giving process (probably because the user is particularly interested in how to balance her diet, which is the topic being discussed). Also, she shows a warmer social attitude as a consequence of a successful dialogue adaptation of the ECA interaction style and overall behavior.

## 6 Conclusions and future work

In this paper we sketch an algorithm for adapting advice-giving dialogues to the user interpersonal stances.

We ground our approach on a combination of methods for social attitude and engagement modeling defined in previous research [23,26]. These methods aim at inferring knowledge about user affective states, basing on the observation of user behavior during the interaction. Linguistic, contextual and conversational features are exploited to infer knowledge about cognitive and affective ingredient of the user state of

mind. In particular, on one hand we employ Dynamic Belief Networks for the dynamic modeling of the user social attitude evolution during the interaction; on the other hand, Hidden Markov Models are employed for the classification of users according to their level of engagement in the advice-giving task.

The recognition methods discussed in this paper have been validated in our previous research with quite satisfying results [22, 23, 26].

Though, open problems still remain. In particular, more research is needed on how to define and adapt the dialogue strategy to the inferred user goal, according to the engagement she shows during the interaction. An ongoing set of evaluation studies will suggest us how to refine the behavior and interaction style adaptation rules of our ECA [31], according to the social attitude shown by the user.

In our future research we plan to conduct evaluation studies to further investigate on how the knowledge inferred from the observation of the user behavior can be successfully employed to adapt the interaction in a successful way.

## References

1. J. Cassell, and T. Bickmore, Negotiated collusion: modelling social language and its relationship effects in intelligent agents. *User Modelling and User-Adapted Interaction*, 13, 1-2, 2003.
2. Lee, E-J., and Nass, C., 1999. Effects of the form of representation and number of computer agents on conformity, in: *Proceedings of CHI*, 238-239.
3. F. de Rosis, N. Novielli, V. Carofiglio, A. Cavalluzzi and B. De Carolis, 2006. User Modeling And Adaptation In Health Promotion Dialogs With An Animated Character. *International Journal of Biomedical Informatics*, 514-531
4. Bickmore, T., Cassell, J. (2005) "Social Dialogue with Embodied Conversational Agents" In J. van Kuppevelt, L. Dybkjaer, & N. Bersen (eds.), *Advances in Natural, Multimodal Dialogue Systems*. New York: Kluwer Academic.
5. R. E. Petty and J.T. Cacioppo. The Elaboration Likelihood Model of Persuasion. In L. Berkowitz (Ed.), *Advances in Experimental Social Psychology*. New York: Academic Press, 19, pp. 123-205, (1986)
6. A. Batliner, S. Steidl, C. Hacker, E. Noth, and E. Niemann, Private emotions vs social interaction: towards new dimensions in research on emotions. In the Proc. of the Workshop on "Adapting the interaction style to affective factors". S. Carberry and F. De Rosis (Eds), July 2005.
7. W. Bosma, and E. André, Exploiting emotions to disambiguate dialogue acts. *Proceedings of the International Conference on Intelligent User Interfaces*. Island of Madeira, 2004.
8. C.M. Lee, S.S. Narayanan, R. Pieraccini, Combining acoustic and language information for emotion recognition. *Proceedings of ICSLP*, 2002.
9. D. Litman, K. Forbes, S. Silliman, Towards emotion prediction in spoken tutoring dialogues. *Proceedings of HLT/NAACL*, 2003.
10. A.J Gill, and J. Oberlander, Taking care of the linguistic features of extraversion. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, 2002.
11. Paiva, A. (Ed), 2004. Empathic Agents. Workshop in conjunction with AAMAS.
12. Rettie, R., 2003. Connectedness, awareness and social presence, in: *Proceedings of PRESENCE*, online proceedings.
13. Polhemus, L., Shih, L-F and Swan, K., 2001. Virtual interactivity: the representation of social presence in an online discussion. *Annual Meeting of the American Educational Research Association*.
14. Blascovich, J., 2002. Social influences within immersive virtual environments, in: R. Schroeder (Eds.), *The social life of avatars*. Springer-Verlag, London, 127-145.
15. Andersen, P.A. and Guerrero, L.K., 1998. *Handbook of Communication and Emotions. Research, theory, applications and contexts*, Academic Press, New York.
16. C. Sidner and C. Lee. An architecture for engagement in collaborative conversations between a robot and a human. MERL Technical Report, TR2003-12 (2003)
17. A. Pentland. Socially Aware Computation and Communication. *Computer*, 38, 3, 33-40 (2005).
18. M. G. Core, J. D. Moore, and C. Zinn, The Role of Initiative in Tutorial Dialogue, in: *Procs of 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary, April (2003)
19. J. Prochaska, C. Di Clemente and H. Norcross. In search of how people change: applications to addictive behavior. *American Psychologist*, 47, 1102-1114, 1992.
20. G. Clarizio, I. Mazzotta, N. Novielli and F. de Rosis: Social Attitude Towards a Conversational Character. In *Procs. of the 15th IEEE International Symposium on Robot and Human Interactive Communication. RO-MAN 2006*.
21. Swan, K., 2002. Immediacy, social presence and asynchronous discussion, in: J. Bourne and J.C.Moore (Eds.): *Elements of quality online education*. Vol3, Nedham, MA. Sloan Center For Online Education. 5.
22. F. de Rosis, A. Batliner, N. Novielli and S. Steidl, 'You are Sooo Cool, Valentina!' Recognizing Social Attitude in Speech-Based Dialogues with an ECA. In A. Paiva, R. Picard and R. Prada (Eds): *Affective Computing and Intelligent Interaction*, Springer LNCS, 179-190, 2007
23. V. Carofiglio, F. de Rosis and N. Novielli: Dynamic User Modeling in Health Promotion Dialogs. In J. Tao, T. Tan and R. W. Picard (Eds): *Affective Computing and Intelligent Interaction*, Springer LNCS 3784, 723-730, 2005
24. Jensen, F.V.: *Bayesian Networks and Decision Graphs*. Springer (2001).
25. Cooper, G.F. and Herskovitz, E.: A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9 (1992).
26. A. Martalò, N. Novielli and F. de Rosis: Attitude display in dialogue patterns. In *Proceedings of AISB'08, Symposium on 'Affective Language in Human and Machine'*, 2008
27. A. Stolcke, N. Coccaro, R. Bates, P. Taylor, C. Van Ess-Dykema, K. Ries, E. Shriberg, D. Jurafsky, R. Martin and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26, 3 (2000)
28. E. Charniak, *Statistical language learning*, MITPress (1993)
29. L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In: *Procs of the IEEE*, 77,2, 257-286 (1989)
30. N. Novielli, C. Strapparava. Supervised and Unsupervised Recognition of Dialogue Acts, in press
31. V. Carofiglio, B. De Carolis, I. Mazzotta, N. Novielli, S. Pizzutilo. Towards a Socially Intelligent ECA, submitted

# Capturing Users' Preferences and Intentions in a Semantic Search System

Caio Stein D'Agostini, Renato Fileto

Post-Graduate Program in Computer Science, Federal University of Santa Catarina

PO Box 476, Florianópolis, SC, 88040-900, Brazil

e-mail: {csd,fileto}@inf.ufsc.br

**Abstract**—A shared representation of knowledge, such as an ontology, cannot capture the viewpoints and preferences of individual users. Praesto is a semantic search system that keeps track of the the user's preferences from his interactions with the system and uses this information on trying to capture the user's intentions in subsequent searches. The user's preferences are represented as his or her context relative to a shared knowledge base (KB), which is used to describe the contents to be searched for. Praesto uses this contextual information to estimate the probable users' intentions, for disambiguating and semantically expanding keyword based searches, in order to provide results that are relevant for the evolving user's interests.

**Index Terms**—context, ontology, semantic search, disambiguation.

## I. INTRODUCTION

Keyword based search is an essential feature of many information retrieval (IR) systems. Traditional implementations of IR systems retrieve resources (text, images, etc) whose descriptive metadata or contents lexically match the keywords provided by the user. However this process is fundamentally flawed when applied to large numbers of users [1], because it does not consider differences among users' vocabularies, neither takes into account semantic relations in these vocabularies, such as synonym, homonym, holonym, and meronym. Limitations can arise even in systems using ontologies and knowledge bases (KBs) to deal with semantic issues, since they do not address how individual users organize their subjective knowledge and how the user interests evolve with time.

While objective information is used to define the parts of the knowledge that are shared by all users [5] (such as common concepts, instances and their semantic relationships described in an ontology and an associated KB), the subjective information includes aspects that are specific to each user, such as individual preferences, cultural background or previous knowledge [3], [1], [2], [5]. In other words, subjective information depends on how a user perceives the objective part of the knowledge and what are his interests in a given time. Ontologies are not suitable for representing subjective information, because they compromise with the majority of the users to represent their objective information.

This work presents a proposal for Praesto, a semantic search system that captures individual users' preferences in order to improve the relevance of the search results for individual users. The preferences of each user are gradually collected from his interactions with the system, especially when posing queries

and browsing results. On the first time a user mentions a keyword, that keyword is looked for in the collection of names of concepts and instances from a KB used to annotate the contents to be searched for, in order to find the descriptions of the possible meanings for that keyword. When there are ambiguities, Praesto asks the user to choose one or more denotations of the keyword that are relevant for him or her, and observe the behavior of the user when browsing results, in order to learn his interests for particular meanings. Then it can provide more precise results for future queries using the same keyword, by taking into account the evolving preferences of a user, represented as his/her context relative to the KB.

Praesto maintains the context of each user in a weighted graph whose nodes refer to specific meanings of keywords previously mentioned by the user. The users' context provides subjective information to automatically disambiguate and semantically expand subsequent queries. After each user interaction, Praesto updates the user's context, in order to keep track of his or her evolving interests. Thus, the maintained context information is an indicative of the user's intentions towards a set of keywords in a particular time, helping to reduce the amount of user's interaction necessary for capturing his or her intentions in each keyword based query.

This paper describes some details of our proposal in a case study about the recovery of articles from Wikipedia, though the system can be customized for particular domains by changing the domain specific parts of its ontology and associated KB. We present domain independent constructions to represent the context of each user, and processes based on the Ant Colony Optimization (ACO) Meta-Heuristic to capture and use the context information when answering keyword based searches.

The remainder of the paper is organized as follows. Section II presents the case study scenario. Section III describes the architecture of the proposed semantic search system. Section IV presents the ACO based processes to capture and use the context information in semantic searches. Section V presents the current implementation and planned experiments. Some related works are presented in Section VI and Section VII closes the paper with some final comments.

## II. CASE STUDY SCENARIO

The case study used in this paper is constructed around DBpedia <sup>1</sup>, a dump of Wikipedia's contents described with

<sup>1</sup>www.dbpedia.org

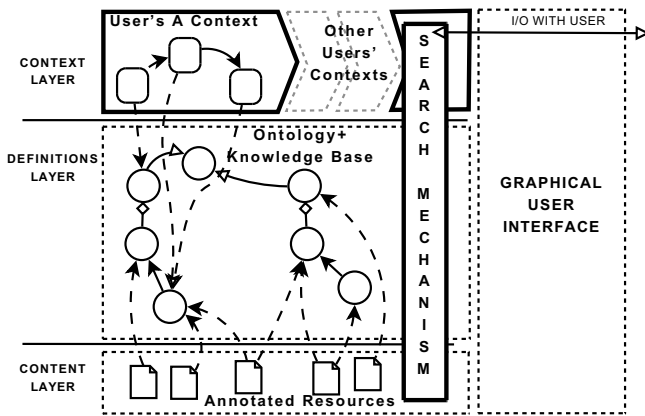


Fig. 1. The architecture of the Praestro system.

an ontology and an associated KB, that were built using Wikipedias's contents. We selected for our experiments a total of 537,295 distinct instances from the DBpedia KB, based on their types (*Organization, MusicGenre, Person, Place*). These instances are looked for based on their property *name*. Take as an example the keyword São Paulo. The Wikipedia's disambiguation page for this keyword lists, among others, the following denotations:

- São Paulo – one of the states of Brazil;
- São Paulo – the capital city of São Paulo state;
- São Paulo – a small island of the St Paul's Rocks;
- São Paulo Futebol Clube – a soccer team.

A SPAQL query for this keyword, searching for instances and concepts whose property *name* has the value São Paulo, returns several results. For the given filtered dataset, there are two concepts referring to distinct denotations of this keyword: *city* and *soccer-team*. Suppose a scenario where the user is searching for São Paulo with the intention to find information about the international airport for traveling to São Paulo the city, which is actually located in the nearby city of Guarulhos. The disambiguation between the possible meanings of São Paulo could be done using information from previous searches. For instance, if the user had previously searched for *airport*, which relates to *city*, the system could return results related to São Paulo, the city, and *airport*. In other words, after some iterations of the user with the system, the context can be used to disambiguate the query, and semantically expand the search as well, in such a way that results about the airport at Guarulhos are considered relevant.

### III. THE CONTEXT-DRIVEN SEMANTIC SEARCH SYSTEM

Praesto is an under-development semantic search system that captures users' context information in order to provide more relevant results for each user query. Figure 1 presents a high-level architecture of the system. This architecture couples the the user's context [15] to the knowledge base (KB) used to annotate the resources. The KB, with instances of an ontology and semantic relationships among them, provides objective

definitions for subjects in one or more domains (definitions layer). Those definitions are used to semantically annotate the resources in the content repository, which is the lowest layer of the architecture (content layer). This paper is focused on the management and use of the context information (context layer), which copes with the subjective information relative to each individual user to represent his or her preferences. This layer keeps the user's preferences by linking the user's vocabulary (collection of keywords that he/she has provided to the system) to specific denotations of these keywords described in the KB, using information collected from previous interactions with the system. The system uses such information for guiding the search process according to the user's interests.

#### A. Representing the Users' Context

Context, for this paper, can be defined as a set of elements related and relevant to subjects from a domain [6]. The context plays an important role in communication [7], as it helps to create consensus between a speaker and a hearer interpreting information [5], [3], thus influencing communication [13].

The individual preferences of each user are represented in this work as a context relative to an ontology and an associated KB, with instances of the concepts of that ontology. Specific meanings of keywords mentioned by the user during his interaction with the system gives rise to topics in his particular knowledge view. Each topic refers to a *term*, i.e., a particular meaning of a keyword ever mentioned by the user. For simplicity we consider that the KB embodies its associated ontology. Thus, a *term* refers to either a concept or an instance of a concept.

The construct selected to represent the user's context in this work is the graph-based [2], [8], [9], [10], [11], [12], [13], [14], [19], [20], because it is the most customizable and flexible among the ones that we have studied [4]. The context of each user is maintained in a weighted graph  $G(T, A)$ . Each node  $t \in T$  is called a topic and refers to a specific denotation of a keyword provided by the user, by pointing to its description in the KB. The weight of  $t$  represents the level of interest of the user's for the respective denotation. Each edge  $a \in A$  refers to an association between topics. The weight of an edge  $a$  represents the level of user's interest for the respective association. Both topics and associations have their *weights* normalized in the interval  $[0, 1]$ . The weights indicate the level of interest of the user for particular topics and associations. A weight value of 0 indicates that, most likely, the respective topic or association is not relevant to the user. On the opposite, 1 indicates that the user has the highest level of interest for the respective topic or association.

A topic is a triple  $t(\text{name}, \text{term}, \text{weight})$  that associates a word *name* from the user vocabulary (provided keywords), to a *term* from the KB  $KB$ , with a *weight* in  $[0, 1]$ .

Two topics can have the same *name*, but they cannot share the same *term*; if they share the same *term*, they cannot share the same *name*. These conditions represent synonym and homonym, respectively. Also, the sum of the weights of the topics with the same *name* must be 1. It enables the system

to compare the user preferences for different denotations of the same *name*, helping to solve ambiguities.

The relations between topics are represented by associations. An association is a triple  $(origin, target, weight)$ . An association from the topic *origin* to the topic *target* is represented by  $origin \rightarrow target$ . The existence of an association between two topics is independent of the existence of a relation between the respective terms in the KB.

The system uses the *weights* of the associations to identify which of the associations departing from a single topic are the most relevant. For that, the sum of the *weights* of all the associations originating in a same topic is equal to 1.

#### IV. THE SEARCH PROCESS

This section describes the process to take advantage of the weighted topic graph (TG) when processing searches and to keep the TG aligned with the user's interests as he or she interacts with the system. This process uses the Ant Colony Optimization (ACO) Meta-Heuristic, which is based on the behavior of foraging ants [16], [17], [18]. Those insects communicate through pheromone trails left in the paths they traverse while searching for food. These pheromone trails are represented in our solution by the weights in the TG, used to guide the search processing.

During the first phase of the ACO, virtual ants are created and sent to find possible solutions for the search, departing from the TG nodes referring to desired denotations of the keywords (the nodes with the highest weights among those labeled with some of the keywords). During this phase, the virtual ants tend to follow paths that produced good results before and thus have stronger pheromone trails (higher weights of topics and associations involved). When a virtual ant finds a solution, it stores the steps it took on its search. In the following phase the system attenuates the relevance (weights of the topics and associations) of paths stored in previous interactions, meaning that, in the future, the ants will have a lower tendency to follow those paths. Finally, in the third phase, the solutions found by each virtual ant are evaluated. Based on this evaluation, the system updates the pheromone trails (weights of the nodes and associations in the paths traversed to find the relevant results).

We adapted the ACO meta-heuristic to this work in a process with two major phases *lookup* and *maintenance* (which includes attenuation, for simplicity), as presented in Figure 2. Praesto follows this process for each searched keyword. After looking up for the keywords in the TG, and in the KB for keywords not present in the TG, it attenuates the weights of topics and associations in the graph, diminishing the importance of older interactions, and updates the graph, according to the keywords used in the search and the feedback received from the user when he/she explicitly chooses denotations or results associated to specific denotations. The detailed processes for each one of these phases are explained in the following subsections.

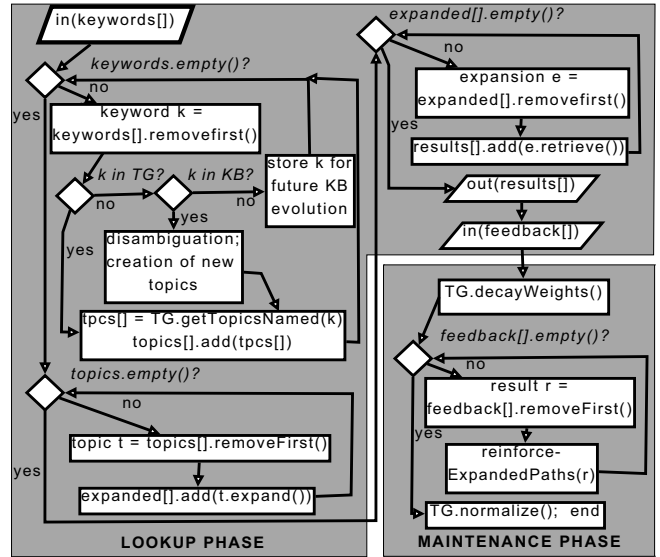


Fig. 2. Praesto's search and TG update process

##### A. The Lookup Phase

The lookup is the first step that Praesto executes. This step receives the user's query as a set of non-exclusive keywords, i.e., connected with the *OR* operator, meaning that the system searches for each keyword independently of the others. Extra-processing, such as identification of composite words or removal of stop-words, if needed, is done previously, being out of the scope of this work.

Praesto searches the TG for topics lexically matching each provided *keyword*. If no topic matches a *keyword*, then Praesto searches for matches in the labels of instances in the KB. When there is ambiguity, i.e., more than one term in the KB matches a *keyword* not present in the user's TG, it is necessary to ask the user what denotation(s) matches his intention on posing that *keyword*. The system does this questioning by presenting to the user all the instances labeled with *keyword* in the KB, along with their related concepts, so that the user can choose the relevant terms. For each instance indicated by the user, the system creates a new topic, with *name* = *keyword* and whose *term* is a reference to the indicated instance from the KB. The *weight* of each inserted topic receives the value  $1/quantity$ , where *quantity* is the number of created topics. This indicates the average relevance of the topic to the user.

It is also possible that there is no match for the keyword, either on the TG or in the KB. This situation is outside this work's current scope, as well using the TG for searching for concepts. Currently, the system warns the specialist in charge of the ontology and the KB about the problem, for their future evolution. Having looked for all the keywords and created new topics or warned the knowledge specialist when necessary, the search proceeds.

After identifying (and possibly creating some) relevant topics, i.e., topics matching the keywords, Praesto starts an

*expansion* departing from each relevant topic. An *expansion* refer to the path traversed in the graph by a breadth-first search for other relevant topics for the particular search and user (and resources annotated to their corresponding denotations). The search is oriented by the *weights* of the topics and of their associations. Each expansion also has its *relevance*. The *relevance* of an expansion is initially the *weight* of the topic that originated the expansion. Each time an association is traversed the topic in the target of that association is added to the expansion the *relevance* of the expansion is adjusted, in order to reflect the *weight* of that association and the *weight* of the topic reached through this association.

The *relevance* of topics and associations is used to decide which paths to follow in the breadth-first search. When the *relevance* of an association is lower than a minimum threshold, the search skips that association and tries the next one. The same applies to the *relevance* of the topic reached by an association. Thus the expansion can be interrupted. The expansion can also be interrupted when its path reaches a maximum depth from the starting topic.

Once the *expansions* are done, the system retrieves the resources annotated by the *terms* from the topics traversed by the *expansions*. Those resources are presented to the user, as results ordered according to their *relevance*, calculated from the *relevance* from the *expansions* that retrieved them.

### B. The Maintenance Phase

As the time goes on, user's interests may change. Topics that were once relevant to the user might not be anymore. Those changes must be reflected in the TG in order to keep track of the user's interests. This is important not only in the long term, but also during user's interactions with the system. The interaction of the user with the search system on posing queries and receiving results referring to the knowledge and information contents can change his/her perception of the searched subject, inducing more changes in his/her interests. Thus, the first step of the maintenance phase is to attenuate the weight of topics and associations used in previous searches, in order to give room to increment the weights of topics and associations used in the current interaction.

The passage of time is applied to the user's context whenever the users provides feedback for search results. When this happens, the *weights* from all the topics and associations in the user's graph are decayed. The intensity of the decay depends on the parameter  $\delta \in [0, 1]$ , where 0 implies the lack of any past memory and 1 indicates that the user's past interests will continue relevant forever.

After the attenuation of old trails, the system proceeds to update the TG according to the feedback provided by the user. For each result  $r$  marked as relevant, the system retrieves the set  $Exp_r$  of *expansions* that resulted in  $r$ . Each  $exp \in Exp_r$  has its path traced back and each association  $(t_i, t_{i+1})$  in the path is reinforced.

The system also increases  $weight_{t_0}$  by 1 when  $name_{t_0}$  matches a *keyword* provided by the user. Thus the system stores that the user is interested in  $term_{t_0}$ , thus enabling the

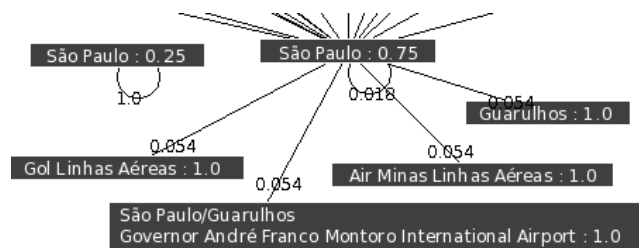


Fig. 3. Topic graph (TG) after some search iterations for São Paulo.

disambiguation of future searches for *keyword*. The system also reinforces all the associations  $(t_0, t_{label})$ , creating them when necessary, where  $t_{label}$  is any topic whose associated *term* is used to annotate *result*. This enlarges the number of topics in the TG which might be relevant to the user. The *names* of the newly created topics provide the system with information about new words that the user might learn by browsing the results. In case these newly created topics are not referenced by the user, they are eventually removed from the TG, by attenuation.

## V. IMPLEMENTATION AND EXPERIMENTATION

The current Praesto prototype implements the three layers of its architecture, using the DBpedia corpus as the KB. This prototype also implements the process described above to use and maintain the contexts of individuals as they interact with the system. Figure 3 shows part of a topic graph (TG) resulting from a sequence of searches for São Paulo, where the user associates São Paulo to *city* (though the chosen results are about the closest international airport). The topic São Paulo is not only associated to that airport, but also to topics related to that airport, such as airlines and its location (*city* of Guarulhos). Despite the amount of searched data, the graph remains efficiently processable by the system.

Praesto's uses subjective information to improve the results' relevance for individual users. Thus, we consider that it is not sufficient to use just pre-existing benchmarks that consider only measures like precision and recall for analyzing its effectiveness. In order to collect more representative measures of the system's benefits, empirical experiments are planned. These experiments will log the user's interactions with the system and measure the gradual increment of precision and recall as the system collects subjective information from the user. Furthermore, we plan to measure the possible reduction in the average number of interactions required to disambiguate queries and the total time spent on posing searches and browsing results.

Those expected benefits are already observable on an under-development prototype of Praesto, but we consider that they should be measured on a version better prepared for the final users; this depends on a proper graphical interface capable of providing a quality visualization of the results. This interface is being developed based on some users' reports of their experience with the current version.

Table VI: Comparative Table

Work	Obj.	Subj.	Status
Graupmann et al. [9]	✓		Impl.
Park, Cheyer [2]	✓	✓	Idea.
Michlmayr et al [12]		✓	Impl.
Aleman-Meza et al [8]	✓		Impl.
Mani, Sundaram [13]	✓	✓	Impl.
Challam et al [10]	✓	?	Impl.
Vallet et al [19]	✓	?	Impl.
Sieg et al [11]	✓	?	PImpl

## VI. RELATED WORKS

This section analyses some related works that take context information into consideration during the search process. Every work is evaluated based on the information published by their authors.

The evaluation criteria are *Objective (Obj.)*, indicating if the work represents the objective part of the knowledge; *Subjective (Subj.)*, indicating if the work has support to capture and use the subjective part of the user's knowledge ('?' indicates some degree of support); and *Status*, indicating if the solution is implemented (Impl), partially implemented (PImpl.) or does not explain how to realize it (Idea.).

According to those criteria, listed in Table VI, only [13] has both characteristics (Subj. and Obj.) and is implemented. It uses a graph for representing the context, but it is constructed directly over ontology's concepts (they are the graph's nodes). It is also targeted to multimedia retrieval and uses different context graphs for each medium (images, sounds, text, etc), which is a different focus than the one from our work.

## VII. CONCLUSION AND FUTURE WORKS

This paper presents a proposal for keeping track of the user's preferences from his interactions with a semantic search system, and use this information to provide search results aligned with the user's interests. The main contributions of this work are: (i) the architecture based on three layers of knowledge, separating specific user's preferences from shared knowledge; (ii) a formal structure to represent the user's context relative to a shared KB; (iii) algorithms for updating the user's context after each interaction with the system; and (iv) ACO based methods to improve the relevance of search results using an individual context.

This proposal is built on the premise that the user's preferences do not change frequently. However, the user can indicate a change of interest at any time, by asking the system to show the possible denotations present in the KB for a given keyword and choosing some of them. Conversely, when there is not enough information in the user's context to solve ambiguities for a keyword, the system asks the user to choose among the possible denotations for that keyword. When a keyword is not found neither in the user's context, nor in the KB, the system feeds a repository of unknown keywords for future extension of the shared KB.

Though there are not enough experimental data to validate the benefits of our proposal yet, controlled executions on small

data collections indicate promising results, which we expect to confirm with the planned empirical experiments. Depending upon the proposal's effectiveness, other experiments and extensions of our proposal can be considered. One possibility is to allow any keyword in the context graph, even those that are not present in the KB. It can increase the adaptability of the search system and contribute for the KB evolution.

*Acknowledgments.*: This work is supported by CNPq (grant 48139212007-6) and FEESC. Special thanks for the help from Grupo-BD/UFSC, LISA/UFSC, LaPeSD/UFSC, professors Mário Dantas and Fernando Gauthier.

## REFERENCES

- [1] Tirri, H.: Search in Vain: Challenges for Internet Search Computer, IEEE Computer Society, 36, pp. 115-116 (2003)
- [2] Park, J., Cheyer, A.: Just For Me: Topic Maps and Ontologies. In: Lecture Notes in Computer Science, 3873, pp. 145-159 (2006)
- [3] Degler, D., Lewis, R.: Maintaining Ontology Implementations: The Value of Listening. In: Extreme Markup Languages 2004, pp. 200 (2004)
- [4] D'Agostini, C. S., Fileto, R., Dantas, M. A. R., Gauthier, F. A. O. . Contextual Semantic Search Capturing and using the User's Context to Direct Semantic Search. In: 10th Intl. Conf. on Enterprise Information Systems (ICEIS), pp. 154-159 (2008).
- [5] Naeve, A.: The Human Semantic Web-Shifting from Knowledge Push to Knowledge Pull. In: International Journal of Semantic Web and Information Systems, 1, pp. 1-30 (2005)
- [6] Souza, D., Belian, R., Salgado, A. C., Tedesco, P.: Towards a Context Ontology to Enhance Data Integration Processes. In: VLDB 08, ACM (2008)
- [7] Winograd, T.: Architectures for Context. In: Human-Computer Interaction, Lawrence Earlbaum, 16, pp. 401-419 (2001)
- [8] Aleman-Meza, B., Halaschek, C., IB, A., Sheth, A.: Context-Aware Semantic Association Ranking. In: First International Workshop on Semantic Web and Databases Berlin, Germany, (2003)
- [9] Graupmann, J., Schenkel, R., Weikum, G.: The SphereSearch engine for unified ranked retrieval of heterogeneous XML and web documents. In: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, pp. 529-540 (2005)
- [10] Challam, V., Gauch, S., Chandramouli, A.: Contextual Search Using Ontology-Based User Profiles Conference. In: RIAO2007 (2007)
- [11] Sieg, A., Mobasher, B., Burke, R.: Ontological User Profiles for Personalized Web Search. In: 5th Workshop on Intelligent Techniques for Web Personalization, Vancouver, Canada, July (2007)
- [12] Michlmayr, E., Cayzer, S., Shabajee, P.: Tech Report: HPL-2007-72: Adaptive User Profiles for Enterprise Information Access HP Labs Technical Reports, HP Labs, (2007)
- [13] Mani, A., Sundaram, H.: Modeling user context with applications to media retrieval In: Multimedia Systems, Springer, 12, pp. 339-353 (2007)
- [14] Huang, W., Prie, Y., Champin, P., Mille, A.: Semantic context representation of resources using annotation graph. In: Proceedings of the Eighth International Workshop on Multimedia Information Systems (2002)
- [15] Mangold, C.: A survey and classification of semantic search approaches. In: International Journal of Metadata, Semantics and Ontologies, Inderscience, 2, pp. 23-34 (2007)
- [16] Dorigo, M., Caro, G., Gambardella, L.: Ant Algorithms for Discrete Optimization Artificial Life, MIT Press, 5, pp. 137-172 (1999)
- [17] Shtovba, S.: Ant Algorithms: Theory and Applications. In: Programming and Computer Software, Springer, 31, pp. 167-178 (2005)
- [18] Panait, L., Luke, S.: Learning ant foraging behaviors. In: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9) (2004)
- [19] Vallet, D., Fernandez, M., Castells, P., Mylonas, P., Avrithis, Y.: Personalized Information Retrieval in Context. In: 3rd International Workshop on Modeling and Retrieval of Context (MRC 2006) at the 21st National Conference on Artificial Intelligence (AAAI 2006) (2006)
- [20] Leake, D., Maguitman, A., Reichherzer, T.: Exploiting rich context: An incremental approach to context-based web search. In: International and Interdisciplinary Conference on Modeling and Using Context, CONTEXT, Springer, 5, pp. 254-267 (2005)



# Toward Developing Knowledge Representation in Emergency Medical Assistance through a Ontology-based Semantic Cache Model

Heloise Manica<sup>1</sup>, Cristiano C. da Rocha<sup>2</sup>, José Leomar Todesco<sup>1</sup>, M. A. R. Dantas<sup>1,2</sup>

<sup>1</sup>Post-Graduate Program in Knowledge Engineering and Management (EGC)

<sup>2</sup>Post-Graduate Program in Computer Science (PPGCC)

Federal University of Santa Catarina (UFSC), Florianopolis, SC, 880400-900, Brazil

heloise@egc.ufsc.br, crocha@inf.ufsc.br, tite@stela.org.br, mario@inf.ufsc.br

Michael A. Bauer

Department of Computer Science

University of Western Ontario

London, Ontario, N6A 5B7, Canada

bauer@csd.uwo.ca

## Abstract

*In this article, it is present knowledge based architecture for a mobile emergency medical assistance system, based on the France SAMU model, adopting ontology and mobile computing approaches. The contribution is characterized for providing routines and medical protocol specifications for specialists through the use of their natural language, collecting elements from this language to develop an ontology domain and also employing a semantic cache for an enhanced utilization of mobile devices. A prototype of the proposal was implemented targeting to support specialists during a day-to-day basis considering knowledge engineering aided by mobile computing techniques. These differentiated characteristics from the contribution have proved to be successfully at early experiments utilizing the implemented prototype.*

**Index Terms - Knowledge representation, ontology, semantic cache, emergency medical assistance**

## 1 Introduction

The growth of published biomedical literature has resulted in an increasing number of independent and heterogeneous data sources. As observed in [13], the biomedical language contains many synonymous terms, abbreviations and acronyms that can refer to the same concept. The terminological diversity is producing a high level inefficiency, especially when researchers are

searching for specific issues. Thus, the task to build a common controlled vocabulary can be considered as a grand challenge problem. Such a vocabulary could be developed for different purposes, examples are: automatic capture in free-text records, literature indexing and retrieval, electronic patient records, statistical reports and billing. A formal ontology can be seen as a controlled vocabulary expressed in an ontology representation language.

The knowledge represented in health domain ontology is important to develop of *clinical decision support system* (CDSS). The work in [12] defines a CDSS as a computer based system that helps health-care professionals make clinical decisions. Despite the advantages, CDSSs are not widely used in practice, especially in the emergency management domain [11].

In this work we present an approach to circumvent the problem of knowledge communication and representation in the emergency assistance setting. The proposed approach is based on mobile devices to provide information anytime and anywhere at the point of care. The model contributes to the ontology development and maintenance in the emergency domain. In the context of ontology maintenance, the proposed approach enhances the controlled vocabulary in emergency domain. A Semantic Cache (SC) model was adopted to deal with mobile computing limitations. On the other hand, to demonstrate our contribution, it was designed and implemented a prototype that executes on mobile devices.

This paper is structured as follows. Ontology and germane characteristics of mobile devices are introduced in section 2. Section 3 illustrates the

proposed approach and describes the prototype implementation. In section 4 it is shown an experimental environment and some results of the proposal. Finally, in section 5 we present our conclusions and directions for future work.

## 2 Ontology and Mobile Devices

The first step in the design of a knowledge base is the decision related to how elements are going to be represented. The domain ontology defines classes of concepts and its relationships. It can be used, for example, as a source of a controlled terminology to describe biomedical entities in terms of their functions and disease involvement. Associating research data with ontology terms can provide efficient data search, by querying using terms at different levels within the ontology [14].

The construction of the ontology is seen as a collaborative activity among people with different expertise [15]. The person who provides the knowledge in a given domain is referred to as expert. The ontology implementation may be guided by a knowledge engineer. As [10] notes, a number of different approaches for ontology development have been proposed. A common element in all such approaches is the enumeration of important terms in the domain. Part of the work described in this paper includes the identification of important terms in the emergency assistance domain.

Recent technological advances in mobile computing makes it feasible to consider the development of mobile CDSS to be employed at the point of care. Mobile devices are well known to offer limited computing capabilities in comparison to desktop systems. They have battery and local storage constraints, low resolution display capabilities and limited computational performance to execute complex applications.

The mobile user must contend with operating the mobile devices with its limited resources. Moreover, care must be taken in the development of the application to ensure that the end user can easily find information that is critical. This is especially true in emergency situations where the emergency personnel are often interacting with other persons or patients, and so cannot have all their attention focused on the device.

In addition to issues arising from mobile devices, wireless environments offer low-quality communication when compared to wired networks. Frequent network disconnections and local noise complicates the provision of information to mobile users.

The mobile and network challenges described above are addressed in our research – we consider the use of a semantic cache model. This model, suggested in [9], can enhance query processing and reduce network traffic flow in a mobile environment. Data caching plays a key role in data management since its use can improve system performance and availability in case of disconnection. Further, it can help save battery power in a client-server communication model and mobile users are still able to work using the cached data when the network is unreachable, as it is shown in our experimental tests.

## 3 The Proposed Approach

In this section, we describe our approach using an ontology-based architecture applied to a mobile emergency service. The major idea is to provide emergency personnel with a ubiquitous tool which can increase the level of available knowledge about the procedures. As a result, the tool can hopefully help emergency personnel increase their probability of success in any intervention. For less experienced personnel, the system may also be used to access knowledge to assure that appropriate protocols are being followed.

The proposed approach, shown in Figure 1, enhances the captured knowledge and maintains the ontology updated with new instances in a semi-automatic way. The user is an expert who submits a query by selecting a keyword and some filters from his PCS (*Personal Communication System*).

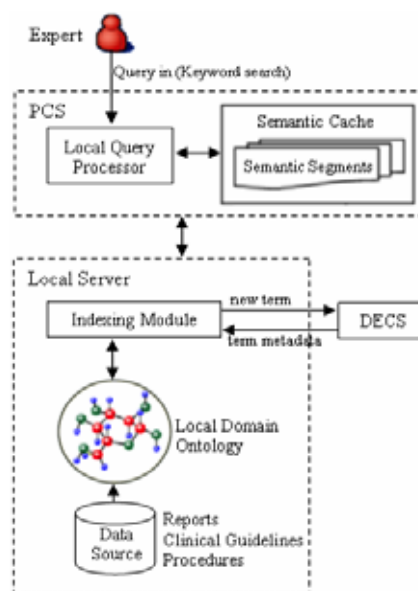


Figure 1. The proposed architecture

The first action of the system is to verify whether a full (or partial) answer for this query exists inside the local module, i.e. inside the semantic cache. In the case that it is necessary to contact the server, the system forwards the query.

Inside the local server, the indexing module correlates keywords from the query with various files (reports, clinical guidelines and clinical procedures, for example) by using the appropriate local ontology emergency vocabulary. When the user initiates a search utilizing a term which can not be found in the local ontology, the local server initiates a search inside the DeCS system [3]. DeCS is a controlled vocabulary developed from the MeSH (Medical Subject Headings U.S. National Library of Medicine).

After acquiring data from the new vocabulary, the index module feeds the local ontology. If the vocabulary does not exist within the DeCS, the unknown term will be stored in temporary area to be subsequently reviewed by experts to determine if this term will (or will not) be included within the ontology.

As a result, the present proposal adopts a strategy which utilizes a semi-automatic approach to feed the domain ontology. If the term occurs inside the DeCS it is immediately inserted in the local domain ontology. In the case that the term does not exist in the DeCS, it is considered later by an expert.

Nowadays, the DeCS cannot be considered as a sufficient element of captured vocabulary for urgency and emergency domain. The main cause for this is that emergency terms are distributed over different categories. Therefore, this represents a challenge difficult for a searching. An example that helps to illustrate this point is a query for the topic emergency burning. A third degree burning is an emergency when it occurred. However, if this occurrence is an accident of six months age and the patient is being treated this is not considered as an emergency burning. As a result, now relevant documents are returned for the query.

If enough time is available for the user, it can refine the query and match the desirable results. This is not the case in urgent or emergency situation. Other well-known challenger is the utilization of specific terms, related to particular linguist and slang characteristics, from a set of specialists.

### 3.1 Domain Ontology Improvement

The proposed ontology considers the Brazilian SAMU mobile health care system (the ideas behind the SAMU can be found in [7]). Emergency experts from the Santa Catarina State [1] were involved with the modeling process, supported by Protégé tool [6], an

open source ontology editor and knowledge-base framework. The process of ontology modeling with Protégé is not covered in this paper.

In our effort to improve the domain ontology we developed a query interface that is shown in Figure 2. This interface allows a user to query through the ontology for relevant documents (e.g., a medical procedures, or clinical guidelines). The specialist enters with the keyword and it is available filters to enhance his/her query. In the Brazilian SAMU, routines (or protocols) are classified as gestures or syndrome. The filter specifies the desirable knowledge, for example, indications which type of material will be necessary, some tips about risk and accidents.



Figure 2. Query Interface

Three scenario examples were conceived to test the prototype. The first scenario considers a key term inside the domain ontology; the second exploits the case where a term exists inside the DeCS and not inside the domain ontology; in the final test the term does not exist inside of the domain ontology or DeCS. The following considerations are relevant to the domain ontology: *i)* files named *ecg\_indicacoes* and *ecg\_materiais* have information referring to utilization and related material of an electrocardiogram; *ii)* the ECG is a synonymous for electrocardiography; *iii)* specialists know that the term EKG is a synonymous for both ECG and electrocardiography.

In the example 1, the user executes a query (Q1) which results in a search of available materials for occurrences of “electrocardiography” as a keyword. This is the simplest case because the term exists inside the domain ontology, therefore the query is executed without problem.

Example 2 is characterized by a similar query from example 1, however now the user seeks for the term “EKG”. This term is synonymous for “electrocardiography”, and the system does not have this knowledge. Because the term does not exist inside the domain ontology the system initiates a search inside the DeCS, where the term “EKG” is found to be synonymous with “electrocardiography”. As a result, the system automatically feeds the domain ontology with this new synonymous term.

In the example number 3, the user enters a query (Q2) searching for material and indications of a keyword “*electrode*”. Because this term does not exist inside the domain ontology or the DeCS, it is inserted as a possible new term into a specific class called *Novo\_termo* (New\_term). The procedure to insert this term inside the domain ontology will be executed manually latter upon a specialist’s agreement that this term is adequate.

As study examples shown, while specialists are executing their queries, used keyword vocabulary is employed to enhance the ontology development. Reliability is an advantage of this approach, because only those elements that were verified by a specialist or DeCS will be inserted in the ontology. Other advantage is that the constant insertion of new terms turns the ontology updated. The action of populate and update the ontology when it is done manually is a time consuming mechanism for specialists.

When a query generates new data for the mobile user, his PCS will also receive a semantic description, which will be used for cache management purposes. The following sub-section describes how this operation is realized.

### 3.2 Semantic Cache

In this sub-section we show how the previous examples were implemented, with special emphasis to the use of the semantic cache proposal. Our semantic cache keeps the results of previously asked queries linked with its semantic descriptions. The semantic information is used to organize and to manage the client’s cache.

In this approach, a *semantic segment* is a tuple  $(S_{ID}, S_{cc}, S_{co}, S_C, S_{T ipo}, S_F, S_{Fq})$ , where  $S_{ID}$  stands for the segment identifier;  $S_{cc}$  the keyword used for query,  $S_{co}$  the synonymous;  $S_C$  a document from query answer;  $S_{T ipo}$  and  $S_F$  both are filters employed in the query;  $S_{Fq}$  a frequency number that indicates the segment utilization.

An answer from any query can be a result from three computational cases: the answer can be only found at a

specific server; the answer is completely inside the SC; part of the answer exists inside the SC. These are essentially the scenarios addressed in the three previous examples. The following revisits them in the context of the SC.

When executing Q1 (from example 1), the system verifies that the cache is empty, thus it sends a complete query to a server, that replies sending *ecg\_indicacoes* and *ecg\_materiais* files. Each file is stored as a new semantic segment.

Suppose a user executes the query Q1 again but using the key-word “ECG”. The system verifies that the answer is inside the cache, because the domain ontology indicates that this term is synonymous with the term “electrocardiography”. Therefore, the query is completely answered with the content existing in the SC, without a required communication with a server. The system updates the frequency of documents’ utilization. In this study, we use the LFU (Least Frequently Used) to delete the data cache replacement.

Finally, suppose a user executes a query Q1, but now a filter *riscos/acidentes* (risks/accidents) is added. The answer for this query is composed of two pieces. The first part, *ecg\_indicacoes* and *ecg\_materiais*, exist inside the cache. The second part *ecg\_riscos* exists inside a server. As a result, only the second part is requested from the server. When the results of this query arrive from the server, they are stored in the cache, together with semantic description. The frequency of the use of *ecg\_indicacoes* and *ecg\_materiais* is updated.

## 4 Experimental Environment and Results

The experimental environment consisted of a server connected to a wired network and a PCS network. The prototype was implemented using the Java language on the server side and J2ME in the client (PCS) side. Mobile devices used in our experiments were Palm Tungsten C, with 200MHz processor, 64 MB memory, 802.11b wireless network interface cards and Palm OS 5.2.1 as an operating system.

The domain ontology was expressed using the Ontology Web Language (OWL) [5], the language recommended by the World Wide Web Consortium (W3C). OWL provides a powerful constraint language for precisely defining how concepts in ontology should be interpreted by machines. The SparQL language [8] and API Jena [4] were utilized for automatic update of the domain ontology.

A traditional object cache application was designed and implemented targeting to draw a comparison with the proposed environment. The object cache model

only re-uses cache data when a query is the same as another in cache.

In the following we present some empirical results adopting the query battery consumption and cache hit rate as metrics for performance measurements. These metrics are especially important for emergency services for a large use of the device and help with available information.

A set of fifteen different queries were conceived aiming to allow an automatic process of test and provide a comparison between applications. A first experiment verifies the battery consumption. This measurement was realized, through the BatteryGraph [2] software package, measuring the two applications before starting and after finishing its execution. Results from this experiment are shown in figure 3.

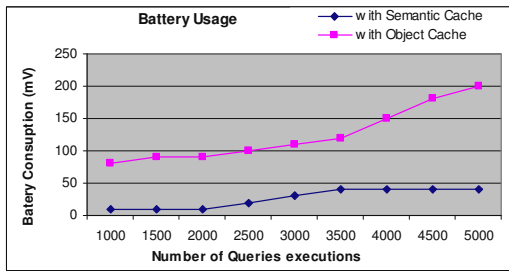


Figure 3. Battery usage

The second experiment was characterized by a mechanism that informs the cache successful access. In other words, it was measured the number of cache hit ratio. Considering all queries it was observed the cache hit ratio as illustrated in figure 4.

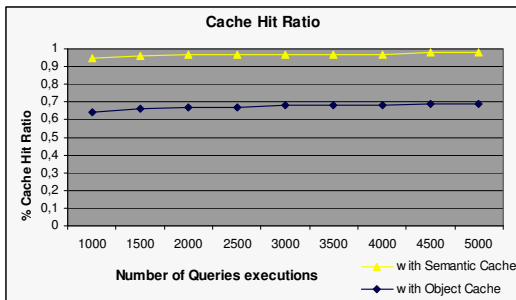


Figure 4. Cache Hit Ratio

## 5 Conclusions and Future Work

In this paper we described the use of ontology technique and semantic cache for a mobile emergency medical assistance system. The proposal architecture was interesting to identify new terms that were used

during an emergency health-care queries. This vocabulary is required to ontology population and to keep the ontological knowledge up-to-date.

Simulation results show that semantic caching achieves a significant performance gain over object caching in mobile environments. The semantic cache is suitable where permanent connection between clients and a server can not be guaranteed, in addition to those cases where crucial functionalities are required without the support of a server.

For future work, we are planning to extend the experiments to examine the *cache hit* rate with different replacement policies.

## References

- [1] SAMU – Santa Catarina, 2009. <http://samu.saude.sc.gov.br>.
- [2] BatteryGraph, 2009. <http://palm.jeroenwitteman.com/>.
- [3] DeCS, 2009. <http://decs.bvs.br/>.
- [4] Jena, 2009. <http://jena.sourceforge.net/>.
- [5] OWL, 2009. <http://www.w3.org/TR/owl-features/>.
- [6] Protégé, 2009. <http://protege.stanford.edu/>.
- [7] SAMU, 2009. <http://www.samu-de-france.fr/fr>.
- [8] SPARQL, 2009. <http://www.w3.org/tr/rdf-sparql-query/>.
- [9] S. Dar, M. Franklin, B. Jonsson, D. Srivastava, and M. Tan. Semantic Data Caching and Replacement. *In Proceedings of the International Conference on Very Large Data Bases*, Mumbai (Bombay), India, pages 330–341. IEEE, 1996.
- [10] D. Jones, T. Bench-Capon, and P. Visser. Methodologies for Ontology Development. *In Proceedings of the Conference of the 15th IFIP World Computer Congress*, London, UK, pages 62–75, 1998.
- [11] W. Michalowski, R. Slowinski, S. Wilk, K. Farion, J. Pike, and S. Rubin. Design and development of a mobile system for supporting emergency triage. *Methods of Information in Medicine*, 44(1):14–24, 2005.
- [12] M. Musen, Y. Shahrar, and E. Shortliffe. Clinical Decision Support Systems. *Medical Informatics. Computer Applications in Health Care and Biomedicine*, pages 573–609, 2001.
- [13] D. Rubin, N. Shah, and N. Noy. Biomedical ontologies: a functional perspective. *Briefings in Bioinformatics*, 9(1):75–90, 2008.
- [14] N. Shah, D. Rubin, K. Supekar, and M. Musen. Ontology based annotation and query of tissue microarray data. *In AMIA Annual Symposium Proceedings*, volume 2006, pages 709–713. American Medical Informatics Association, 2006.
- [15] P. Zweigenbaum, B. Bachimont, J. Bouaud, J. Charlet, and J. Boisvieux. Issues in the Structuring and Acquisition of an Ontology for Medical Language Understanding. *Methods of Information in Medicine*, 34(1):15–24, 1995.

# Specification of a Component-based Domotic System to Support User-Defined Scenarios

Fady Hamoui<sup>1</sup>, Marianne Huchard<sup>2</sup>, Christelle Urtado<sup>1</sup>, Sylvain Vauttier<sup>1</sup>

<sup>1</sup> LGI2P / Ecole des Mines d'Alès  
Nîmes – France  
<First>.<Last>@ema.fr

<sup>2</sup> LIRMM, UMR 5506, CNRS and Univ. Montpellier 2  
Montpellier – France  
huchard@lirmm.fr

## Abstract

*Many studies have been conducted in order to develop systems that respond to user goals in domotic environments. These systems generally offer predefined scenarios corresponding to general goals and enable users to select those they want to trigger. We claim that such behaviors cannot be hardwired: user scenario definition should be supported. In this article, we propose the specification of a component-based domotic system that tackles this issue. This system offers users high level GUIs to define their own scenarios from functionalities of the devices detected in the environment. These scenarios are automatically implemented: components are generated from device descriptors, assembled and the resulting software is run.*

## 1 Introduction

Domotic environments are composed of electrical/electronic devices controlled by a domotic system that uses software and communication technologies to have the devices satisfy user goals. Each device provides control services. Each service in turn offers functionalities. Some devices also emit events that reflect a change of their parameter values. A domotic system can thus be seen as both a set of distributed services and a system that manipulates these services to achieve user goals. Users may simply use existing functionalities or need to combine them in a more complex scenario. Domotic environments can be used in many different ways. As it is not possible to hardwire all possible user scenarios, users must be given the capability to define their own scenarios. Furthermore, scenario integration should be automatic and dynamic, in order not to interrupt system execution [7, 10] and scenario execution must rely on some technical solution that co-

ordinates service executions. Service Component Architecture [18] is a good candidate. It provides a model for the composition of services. Software components are the best means to implement services. They expose interfaces that describe functionalities, each of which represents a service [5]. Components can be dynamically assembled to achieve dynamic service connection [5, 7, 16, 9] without disrupting system execution [6]. Such systems have been developed by industry or academics. They are generally included in a fixed or mobile housing of control that allows to act on home devices such as shutters and lights. Few of them support complex user goals: most are exclusively based on predefined scenarios. Our goal is to specify a self-configurable system that runs (combinations of) services available on nearby devices. This system should seamlessly integrate user-defined scenarios without disrupting its execution. Users should easily express their goal scenarios with a dedicated end-user language [4]. The remainder of this article is as follows. Section 2 lists qualities that we expect from a domotic system. Section 3 describes our domotic system from the user point of view: it shows how users can describe their own scenarios. Sections 4 and 5 further describe our system by respectively providing its meta-model (as its structural view) and its process-oriented two-phased description (as its dynamical view). Section 6 compares to existing proposals while Sect. 7 concludes and draws perspectives to this work.

## 2 Target qualities for domotic systems

Let us consider a domotic environment composed of shutter, radiator and clock devices. The clock, for example, provides a service to set or get time and an event that indicates time change and contains the new time. Users must have the capability to define the following

evening scenario: at 07:00 PM, if the living-room temperature is below 17°C, the shutter should be closed and the radiator turned on at level 6. In order to support such scenarios, domotic systems should have the following qualities [3, 2]. *Decentralization* makes the software spatial structure stick to the physical distribution of devices. It also increases software quality and availability by distributing the load on several units and reducing the impact of failures. *Ability to define goals* allows users to add custom scenarios at any time. *Dynamic evolution* makes the system reactive to changes without impacting service continuity. *Autonomy* limits user intervention to scenario definition: technical steps that implement scenarios are under system responsibility.

### 3 User goal-oriented functions

To meet these requirements, our system is composed of software agents built from software components. Agents are autonomous and collaborative entities. They have a flexible internal structure that allows dynamic (re)configuration through the (re)assembly of components. We identified two types of agents: GUI agents and device control agents (DCAs). GUI agents are a software mediator between devices and users. They enable users to customize the domotic system and define their goals. DCAs are responsible for the detection of devices that are available in the environment and for the execution of user-defined scenarios through their ability to control devices. Users can explicit their goals using services provided by the available devices by either selecting a particular service or defining a complex scenario. To do so, GUI agents make graphical user interfaces that represent the domotic environment available to users. These GUIs are automatically generated from the descriptors of detected devices.

#### 3.1 Service selection

Using the dedicated GUI, users can select a device to display its provided services and select one. Each service in turn offers a set of parameterized operations. Users select such an operation and provide adequate parameter values. The system then invokes the required functionality. For example, the user can select



Figure 1. Service selection GUI

the *clock* to view its provided services. A single service is available that provides the *set time* and *get time* operations. The choice of the *set time* operation enables the user to specify the new time as shown on the simple GUI of Fig. 1.

#### 3.2 User scenario definition

Using the dedicated GUI, users can define new complex scenarios. A scenario is defined by several Event / Condition / Action (ECA) rules [12] that combine various operations. ECA rules enable the coordination of services as they are active (their execution is automatically triggered), express alternatives (with their condition clause) but are declarative (easier to read) and still interpretable [12]. Users must successively define the three clauses of the new rule as illustrated by Fig. 2 for the *evening scenario* example.

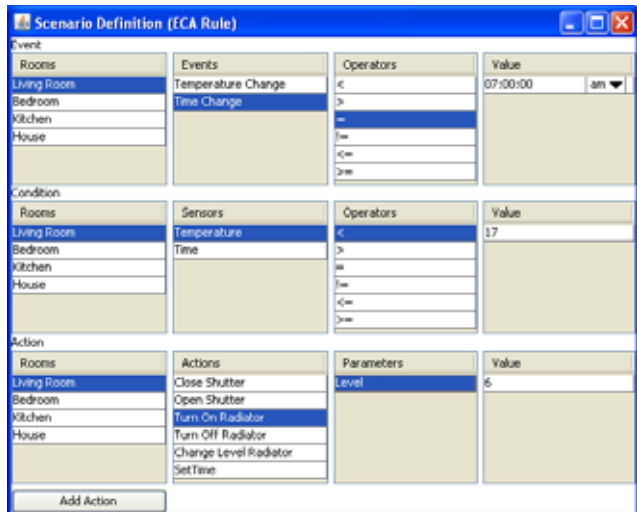


Figure 2. Scenario definition GUI

**Event clause.** An event is a pre-condition for triggering action executions. The GUI displays a list of all available events (the sum of all events that can be emitted by all detected devices) among which users choose the one that suits their needs. In the example, the event is *at 07:00 PM* and is obtained by comparing with the = operator the *07:00 PM* parameter value to the time provided by the *Time change* event of the living room.

**Condition clause.** The condition clause defines in which cases the rule will be triggered. A condition is a boolean function with at least two parameters, provided by either the user or measurement functions offered by sensor devices. Thus, the GUI displays all measure services available. The user chooses such a

service and a comparison operator. He then provides a value to compare to. In the example, the condition is *if the temperature in the living room is below 17°C* where *the temperature of the living-room* is provided as a service by a sensor device, *<* is a comparison operator and *17°C* is a user-provided parameter value.

**Action clause.** The action clause contains one or more service operations that perform actions on devices. The user selects a device, chooses an available operation and, if needed, specifies values for its parameters. In the example, the actions are *the shutter should be closed* (no parameter) and *the radiator should be turned on at level 6* (6 is a parameter value).

At the end of scenario definition, a *coordination descriptor* is generated that contains data relative to the ECA rule. It is stored by the GUI agent and assigned to the DCA that will implement it. The system also contains predefined (or previously defined) scenarios users can execute directly.

## 4 Meta model of the component-based domotic system

This section aims to describe more precisely the proposed system by providing its meta-model. For readability's sake, the meta-model representation is divided into three views. The first view presents our service typology and the correspondence between services and ECA rule clauses. The second view shows what scenarios are and how services that compose a scenario are advertised in the service directory. The last view is devoted to showing how the agents that compose the system are made from software components and component connections.

### 4.1 Service typology

We have identified five service categories (see Fig. 3). *Sensor services* provide measures that come from sensor devices. They are used to provide measures in the condition clause of rules. *Event services* are used in event clauses: they provide events emitted by sensor devices and allow to detect changes in the environment. *Action services* are used in the action clauses: they perform operations on actuator devices thus providing services to users. *Comparison services* are used in condition clauses: they are (mostly predefined) technical services that provide comparison methods for all primitive types. *Coordination services* enable scenario execution. As scenarios are defined by ECA rules, they integrate a rule execution engine.

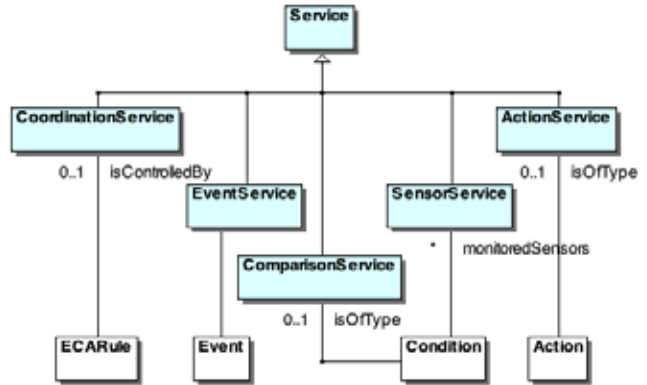


Figure 3. Service typology

### 4.2 Service directory and user scenarios

Agents have access to a service directory that enables inter-agent cooperation without hard-coding the underlying dependencies (decoupling). This directory (see Fig. 4) contains information on:

- *services* offered by devices of the environment (*Service* class). A service has a single type, represented by the *Interface* class. It can be provided by several service providers. Each service provider is bound to concrete component interfaces (often represented as a *lollipop* as shown on Fig. 6), represented by the *FunctionalInterface* class.

- *events* emitted by devices of the environment. They are represented by the *EventService* class as a specialization of the *Service* class. As for general services, an event is of a certain event type and can be provided by several event providers. Each event provider is bound to concrete event interfaces (sometimes represented as a *triangle* as shown on Fig. 6), represented by the *EventInterface* class.

- *parameter types* encountered in operations offered by devices. They are represented by the *ParamTypeInfo* class. Primitive types (numbers, strings, dates, times, etc.) are instances of the *ParameterType* class.

User scenarios (see Fig. 4) are defined by one or more ECA rules that are composed of an event, a condition and of one or more actions. These clause elements refer to the corresponding service advertisements (see Fig. 3) and are further mapped to corresponding concrete component interfaces (events to event interfaces, conditions and actions to functional interfaces) when service providers have been found / chosen for each necessary service. Parameter values defined by users during rule condition or action definition are mapped to parameter type information from the directory. When the condition clause is built from measure services (as



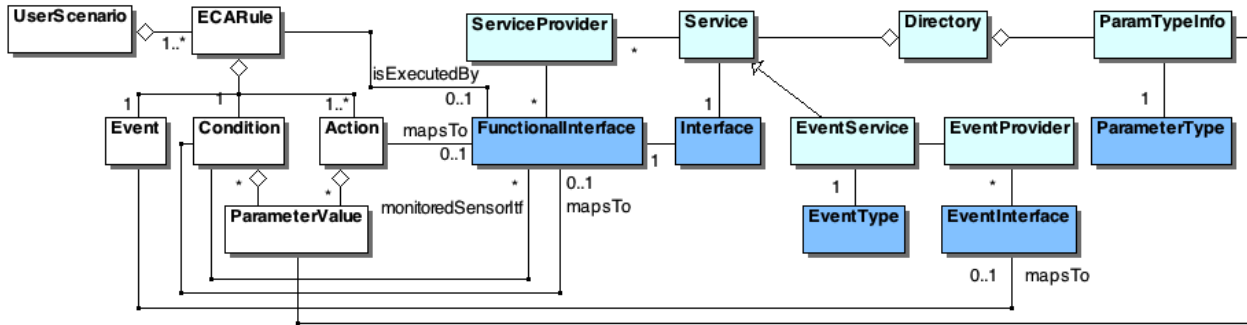


Figure 4. Service directory and user scenarios

results of some sensor service execution) the condition clause of the rule is linked to the functional interface that corresponds to this sensor service.

### 4.3 Component-based software agents

In our proposal, agents are built from components. The meta-model proposed here for component assembly (see Fig. 5) inspires from our previous work on self-assembling components [9, 8]. The two agent categories we have identified in our system (GUI agents and DCAs) specialize general agents made from components (*Agent* abstract class). Each agent has access to a service directory. GUI agents enable users to define their goals in the form of scenarios (stored as their *scenarioList*). DCAs detect services and events provided by available devices and execute scenarios. To do so, DCAs are composed of four types of components that mostly follow the typology of services provided in Sect. 4.1. *Sensor components* retrieve measures provided by sensor devices. *Action components* perform actions on actuator devices. *Comparison components* execute comparison services on primitive parameter types. Finally, *coordination components* control and coordinate the three previous component types to execute a scenario. These components all are generated by DCAs from device descriptors and built-in information on data types. Software components export their requirements and provisions through interfaces, represented by the *ConnectableInterface* class. Two components interact through the assembly of two interfaces, one provided by a component and the other required by the second component. Components export both functional interfaces and event interfaces (modeled as specializations of the *ConnectableInterface* class). Whatever its direction, the type of a functional interface is defined by an interface (*Interface* class) that can be compared to those of Java. These interfaces group operation declarations (modeled by the *Operation* class)

each of which involves any number of input parameter types and at most an output type. Sensor components export one or more provided event interfaces. Whatever its direction, an event interface is typed by an event type. An event interface is a channel through which events are emitted when the value measured by some sensor changes. The event contains the new value. Coordination components export one or more required functional interfaces and an event interface.

## 5 Domotic system dynamics

The dynamics of the domotic system can schematically be decomposed into two phases.

**Self-configuration phase.** This phase consists in the detection of available devices to set up the system and maintains accurate information on devices. Each device is described by a descriptor which contains information on services and events they provide. DCAs download these descriptors and extract the information needed to generate sensor and action components. Then, they advertise information on the services and events provided by each component into the directory. This self-configuration phase executes autonomically at system startup and re-executes periodically to detect device or service addition or removal.

**Self-assembly phase.** This phase translates user scenario definitions into operational component assemblies that implement the scenarios. After a scenario is defined, the corresponding coordination descriptor is sent to a DCA for it to parameterize the rule execution engine of a corresponding coordination component. Then, the coordination component is assembled to the declared sensor, action and comparison components. Once the assembly achieved, the scenario is activated. The coordination component then listens to events, is able to retrieve values from its sensor compo-

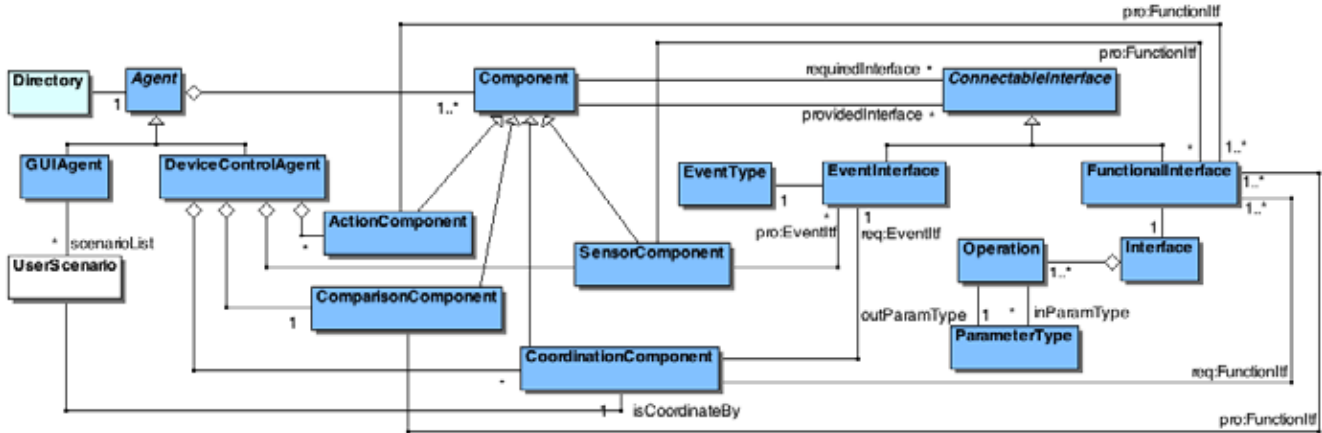


Figure 5. Agent and component typology and component external description

nents, compute the value of the condition with its comparison component and execute the prescribed actions thanks to its action components. The coordination descriptor of the *evening scenario* contains the information necessary for the DCA to parameterize the coordination component and assemble it to the *Clock*, *Radiator*, *Shutter* and comparator components. The descriptor and resulting assembly are presented in Fig. 6.

## 6 State of the art

**Component models.** Sensor Beans (SB) [13] and SOFA's [17] component models are close to ours. They both propose a typology of interfaces and oppose to our approach that relies on a component typology. Our action components nonetheless have interfaces that correspond to *Service* (SB) and *CSProcCall* interfaces (SOFA) while our sensor components have interfaces that correspond to *Event* and *Producer/Consumer* (SB) and *EventPassing* and *DataStream* interfaces (SOFA). Our proposal is thus comparable as for the syntactic richness of interfaces but further adds semantics through a component typology.

**Domotic systems.** Existing domotic systems fit into three categories. *Predefined Scenario Systems* contain centralized systems based on predefined scenarios. In [2, 3, 10], the only capability offered to users is to choose the scenarios they want to execute. The implementation of a scenario generally consists in assembling existing components. [2, 3] provide a slightly more general architecture: new components are generated as bridges, to enable interoperability between various technologies. In our proposal, the components that are generated are not dedicated to satisfying technical purposes but to meeting new user

goals (they encompass some semantics on the system). To conclude, to our opinion, predefined scenarios are not sufficient to cover all possible situations and meet all user-goals: they are not change-resistant as any unforeseen change requires the intervention of an expert user. *Service Control Systems* [1, 11, 14, 19] allow users to control available services. They automatically detect devices in their environment and build a user GUI that lists the services provided by the detected devices. This capability is very close to the service selection GUI provided in our system. The user interacts with the system through this GUI to trigger service executions but cannot define complex scenarios. Among them, [19] nonetheless allows to define simple scenarios as service sequences. *Scenario Definition Systems* enables users to define their own scenarios. [7] offers a tool to define scenarios that is designer-oriented and does not allow runtime scenario definition. Similarly to our proposal, [15] provides users with a GUI for scenario definition and execution. However, scenarios seem restricted to sequences of service calls: they do not propose conditional executions as ECA rules do. Moreover, there is no possibility for users to dynamically define service parameters in their scenario scripts.

## 7 Conclusion and perspectives

In this paper, we described the specification of a domotic system that enables users to define their own goals. The system consists of a set of component-based agents. It automatically detects services and events offered by available devices and includes them in a GUI that represents the environment. Users can use a simple service implemented by a generated component or define a scenario represented by a new (automatically produced) component assembly formed by generated

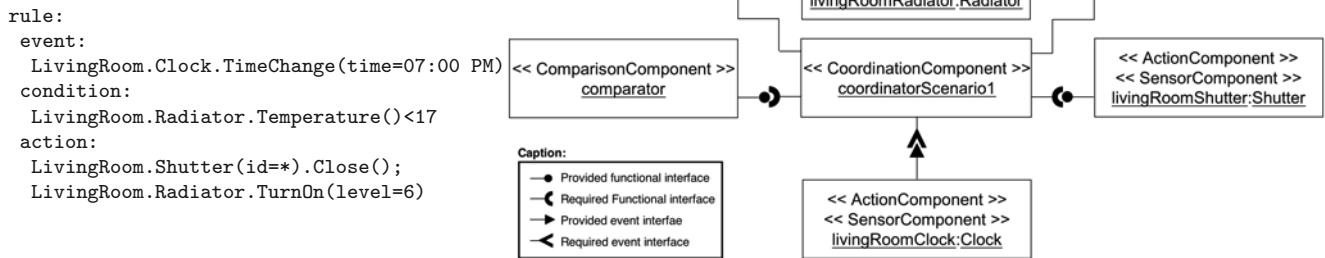


Figure 6. Evening scenario rule descriptor and component assembly

components. The system is under development using the OSGi and UPnP standards<sup>1</sup>. In the future, it will be enhanced with new component types that implement scenario conflict management, fault tolerance policies, automatic service adaptation, etc.

## References

- [1] S. Berger, H. Schulzrinne, S. Sidiroglou, and X. Wu. Ubiquitous computing in home networks. *IEEE Communications Magazine*, 41(11):128–135, Nov. 2003.
- [2] A. Bottaro, A. Gerodolle, and P. Lalanda. Pervasive service composition in the home network. In *21<sup>st</sup> Int. Conf. on AINA*, Niagara Falls, Canada, pp 596–603, May 2007. IEEE.
- [3] J. Bourcier, A. Chazalet, M. Desertot, C. Escoffier, and C. Marin. A dynamic-SOA home control gateway. In *IEEE Int. Conf. on SCC*, Chicago, USA, pp 463–470, Sept. 2006.
- [4] M. Burnett, S. K. Chekka, and R. Pandey. FAR: An end-user language to support cottage e-services. In *Proc. IEEE Int. Symp. on Human-Centric Computing Languages and Environments*, Stresa, Italy, pp 195–202, May 2001.
- [5] H. Cervantes and R. S. Hall. Automating service dependency management in a service-oriented component model. In *6<sup>th</sup> Wkshp on CBSE*, Portland, USA, May 2003.
- [6] Y. Charif-Djebbar and N. Sabouret. Dynamic service composition and selection through an agent interaction protocol. In *IEEE/WIC/ACM Int. Conf. on WI-IAT*, Hong Kong, pp 105–108, Dec. 2006.
- [7] D. Cheung-Foo-Wo, J.-Y. Tigli, S. Lavirotte, and M. Riveill. Wcomp: a multi-design approach for prototyping applications using heterogeneous resources. In *IEEE Int. Wkshp on RSP*, Chania, Crete, pp 119–125, 2006.
- [8] N. Desnos, M. Huchard, G. Tremblay, C. Urtado, and S. Vauttier. Search-based many-to-one component substitution. *Journal of Software Maintenance and Evolution*, Wiley, 20(5):321–344, Sept./Oct. 2008.
- [9] N. Desnos, S. Vauttier, C. Urtado, and M. Huchard. Automating the building of software component architectures. In *3<sup>rd</sup> Europ. Wkshp on EWSA*, Nantes, France, LNCS, 4344:228–235, Sept. 2006. Springer.
- [10] G. Grondin, N. Bouraqadi, and L. Vercoeur. MaDcAr: An abstract model for dynamic and automatic (re-)assembling of component-based applications. In *9<sup>th</sup> Int. Symp. on CBSE*, Västerås, Sweden, LNCS, 4063:360–367, June 2006. Springer.
- [11] H. Ishikawa, Y. Ogata, K. Adachi, and T. Nakajima. Building smart appliance integration middleware on the OSGi framework. In *7<sup>th</sup> IEEE Int. Symp. on ISORC*, Vienna, Austria, pp 139–146, May 2004.
- [12] J.-Y. Jung, J. Park, S.-K. Han, and K. Lee. An ECA-based framework for decentralized coordination of ubiquitous web services. *Information & Soft. Tech.*, 49(11-12):1141–1161, Nov. 2007.
- [13] C. Marin and M. Desertot. Sensor bean: a component platform for sensor-based services. In *3<sup>rd</sup> Int. Wkshp on MPAC*, New York, USA, pp 1–8, 2005. ACM.
- [14] K. Matsuura, T. Hara, A. Watanabe, and T. Nakajima. A new architecture for home computing. In *IEEE Wkshp on WSTFES*, Washington, USA, pp 71–74, May 2003.
- [15] M. Nakamura, H. Igaki, H. Tamada, and K. ichi Matsumoto. Implementing integrated services of networked home appliances using service oriented architecture. In *2<sup>nd</sup> Int. Conf. on SOC*, New York, USA, pp 269–278, Nov. 2004. ACM.
- [16] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing special section. *Communications of the ACM*, 46(10):24-28, Oct. 2003.
- [17] F. Plásil, D. Bálek, and R. Janecek. SOFA/DCUP: Architecture for component trading and dynamic updating. *Int. Conf. on CDS*, pp 43–52, 1998.
- [18] SCA Consortium. Building systems using a service oriented architecture. Whitepaper available from [www-128.ibm.com/developerworks/library/specification/ws-sca/](http://www-128.ibm.com/developerworks/library/specification/ws-sca/) [Last checked 2009-04-30], 2005.
- [19] C.-L. Wu, C.-F. Liao, and L.-C. Fu. Service-oriented smart-home architecture based on OSGi and mobile-agent technology. *IEEE Trans. on SMC, Part C*, 37(2):193–205, 2007.

<sup>1</sup><http://www.osgi.org> and <http://www.upnp.org>.

# Towards Mobility Support in Smart Environments

Daniel Retkowitz      Ibrahim Armac  
Manfred Nagl

Department of Computer Science 3 (Software Engineering)  
RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany  
{retkowitz|armac|nagl}@i3.informatik.rwth-aachen.de

## Abstract

*Smart environments are subject to intensive academic and industrial research. Many of these research projects deal with challenges such as heterogeneity, personalization and context-awareness. However, most of them assume smart environments to be insular places. Considering users visiting different environments in daily life, this assumption becomes unrealistic.*

*Mobile users wish to use personal functionality in their home environment as well as in other environments they visit in daily life. In this paper we describe different realization patterns to implement services for smart environments. The aim is to support personalization of services and mobility of the users. Depending on the application, different realization patterns are preferable. Furthermore, we describe how our prototype implementation supports the different patterns.*

## 1. Introduction

In this paper we describe our approach on supporting personalization and mobility in smart environments, in particular smart homes, which we call *eHomes*. These are environments equipped with devices which are usually connected to a hardware platform called *residential gateway*. This gateway runs software services to realize value-added functionality across multiple devices.

A specific challenge in realizing eHomes is to deal with mobility. One kind of mobility is given when users move from one location to another one (*in-home mobility*). In most cases a *location* is a room in an eHome. However also larger areas of an eHome that comprise several rooms or a part of a larger room can be modeled as one location. Another kind of mobility is given when users move from one eHome to another eHome (*inter-home mobility*). In this case, the term eHome is used

in a broader sense meaning also environments such as a hotel, work place etc. Furthermore, also the mobility of devices and changing user preferences have to be taken into account to support dynamics in eHomes.

Considering inter-home mobility, the important question arises how to support users in personalizing visited environments. For this purpose we pursue a *client side personalization* approach. This approach is based on the assumption that every user carries a smart mobile device which can support the user in personalization tasks. Nevertheless, not all services need to be personalized. Therefore we distinguish *personal services* which adapt their functionality to user preferences and *non-personal services* which provide functionality at a specific location in an eHome or for an eHome as a whole.

Mobility requires a dynamic eHome system that reacts on changes and adapts to the new situation. In our project we developed a configuration approach that especially supports the requirements becoming apparent in dynamic scenarios considering mobility of users and devices. We analyzed different patterns for realizing services in mobile scenarios, where to apply them, and what implications they bring along.

We will describe our approach on supporting personalization and mobility in Section 3. Before that, we introduce our eHome system model in Section 2. In Section 4, we will discuss related work. Finally, we will conclude the paper with a summary and an outlook to future work in Section 5.

## 2. System Model

In future smart environments we assume different usage scenarios. Typical services provide functionality from the domains of comfort, entertainment, communication, security, health care, or time and energy saving. An *eHome service* implements a certain functionality, which is provided either directly to the users of

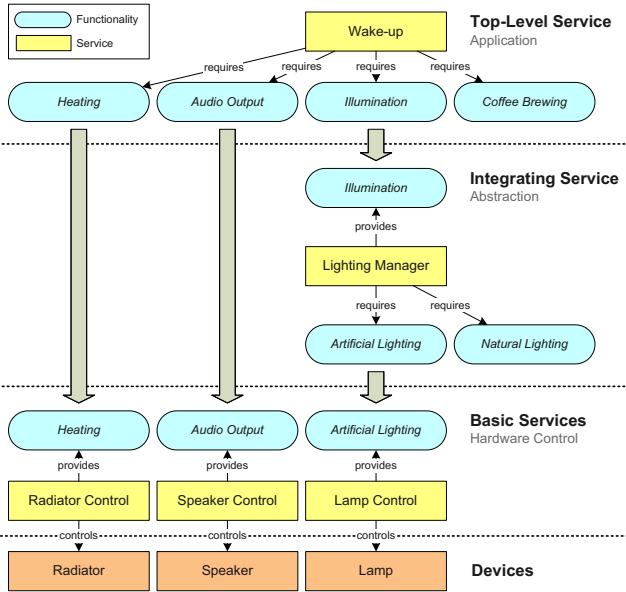


Figure 1. Layered Services

an eHome or to other services. We distinguish between three different service types: top-level, integrating, and basic services. Services may rely on functionalities provided by other services on a lower level of abstraction. This leads to a layered service architecture.

### 2.1. Service Layers

Figure 1 shows a wake-up service as an example. This type of service is called *top-level service* since it provides its functionality directly to the user. The wake-up service requires other functionalities to operate, which have to be provided by services on a lower abstraction level. In this case heating is required to increase the room temperature before wake-up time. Audio output is required to play some wake-up sound or music. Coffee brewing functionality is used to prepare coffee after wake-up, so the person does not have to wait during the brewing process. Illumination functionality is used to slowly increase the illumination level at the location of the person to wake-up. This allows for a comfortable wake-up procedure. Illumination is here controlled by an intermediate lighting manager service, which provides illumination based on artificial or natural lighting. In case there is bright sunlight outside, the roller blinds can be used to control the illumination level. In other cases, especially during the night of course, artificial lighting is used for this purpose. On the lowest level of abstraction basic services are used to provide access to the available hardware in the eHome, e.g. to control radiators, speakers, or lamps.

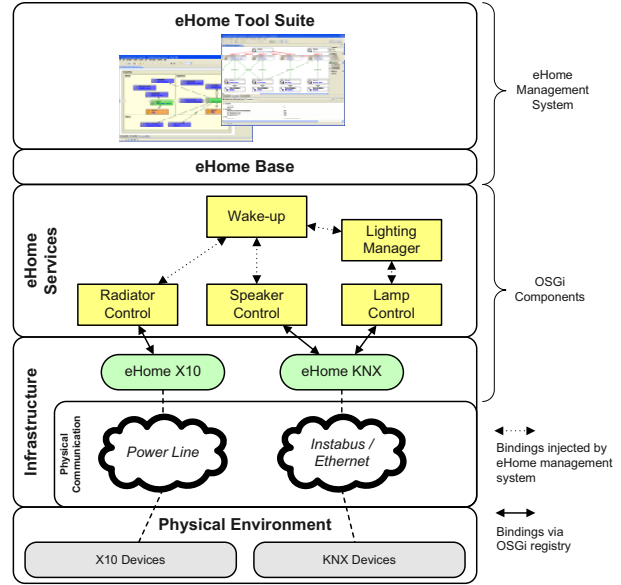


Figure 2. System Overview

A service composition like in the wake-up scenario is depending on the available hardware and the current status of the environment. Since changes occur frequently at runtime, the service composition has to be adaptable. In our approach the configuration of the eHome is managed by a service-oriented middleware running on the residential gateway. In the next section we describe the basic system architecture we apply. Details of the configuration mechanism beyond the scope of this paper are described in [7].

### 2.2. Service Gateway

The *service gateway* is a software platform for executing eHome services. It is running on the residential gateway, the central hardware unit of the eHome, which is in control of all hardware usable by eHome services. This means that the services are not distributed in terms of their execution. Only the device hardware that is controlled by driver services is distributed. Nevertheless, a service can be bound to a specific location. This means that the provided functionalities of that service take effect at this specific location.

The system configuration is also managed centralized by the service gateway. We pursue an approach based on a global view of the eHome and its current environment status. This way global context information can be taken into account for service composition, e.g. the location of persons and devices in specific rooms. Global knowledge of the environment is required for a meaningful service composition in many typical cases.

A simple example is the requirement to bind a resource from the specific location a service is associated to. This requirement can only be formulated if we have a concept of different locations in the first place and if we know which resources are available at this location at a given time.

Figure 2 shows an overview of the service gateway architecture we apply and how the service gateway is connected to the devices in the eHome. The top-most layer is a graphical interactive tool called *eHome Tool Suite* that is used to monitor and administrate the eHome system at setup and during runtime. A service specification editor is also integrated into this tool.

The data model and application logic for managing the eHome system is realized on the underlying eHome Base component. The data model is used as a representation of the current state of the eHome system comprising the physical structure of the eHome but also the dynamic state, i. e. the currently present users and their positions inside the building, the active devices, and the running services and their composition. The application logic of the eHome Base component implements the control capabilities to manage the eHome system configuration, i. e. the composition of services, service parameterization, and other runtime aspects. Furthermore, the deployment of system configurations is performed by this component.

The deployed instances of eHome services are executed on the next layer. This is controlled as described above by the eHome management system, i. e. the eHome Tool Suite and the eHome Base component. Services are composed according to their required and provided functionalities, the current environment status, and the user's specific requirements. Depending on the type of the used hardware a corresponding infrastructure is used for accessing this hardware, e. g. X10 devices or KNX (ISO/IEC 14543) devices. These devices are connected to the residential gateway via the eHome's power line or KNX network infrastructure, respectively.

### 3. Service Realization Patterns

In this section, we will introduce a classification of possible patterns for realizing eHome services, discuss our approach on mobility and configuration support, and describe implementation details.

In Section 2 we have described the different service layers. In this section we will discuss different realization patterns of top-level services regarding their bindings to locations or persons.

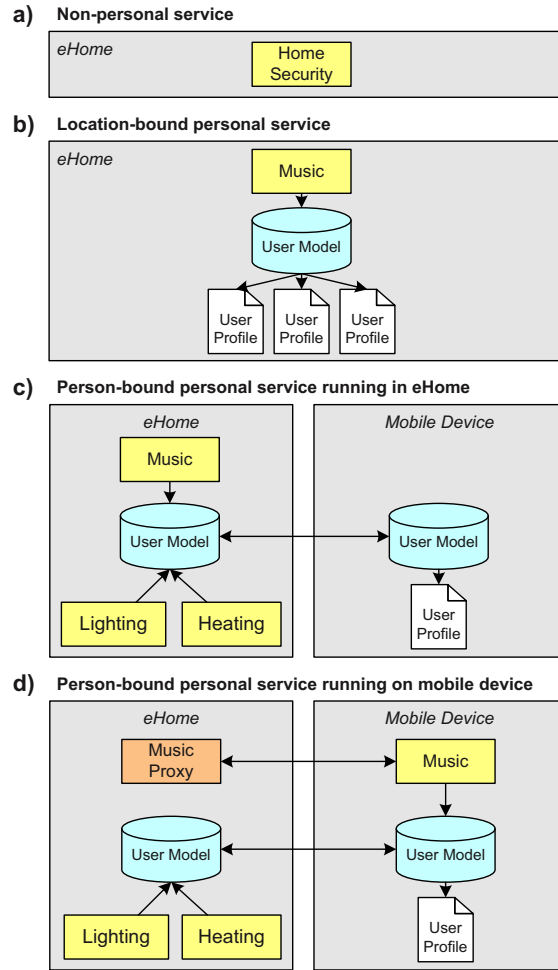


Figure 3. Different realization patterns for eHome services

#### 3.1. Non-personal Services

Non-personal services are usually bound to locations and are not related to any specific person, as depicted in Figure 3a. Such services provide general location-based functionality, e. g. a home security service that detects intrusion or fire and raises an alarm. They obviously do not require any personal data to operate and are only related to spatial context. Any non-personal service is usually bound to a specific location where its functionality will take effect.

#### 3.2. Personal Services

Personal services are related to individual persons and therefore require personal data, provided by user profiles. Besides the spatial context they also relate

to personal context, e.g. a music service depends on the user's location and music preferences. We have developed a *user model* which holds personal data and provides it to personal services by a unified interface. There are different ways to realize personal services.

A *location-bound realization* is similar to the realization of non-personal services in that the service is associated to a specific location. In contrast to non-personal services now the users present at this location are taken into account. Depending on the presence of users the service accesses the user model which provides personal data for the different users, as shown in Figure 3b. Based on user preferences, the service personalizes its functionality for a specific user. However, if the user leaves the service's location and some other user arrives the service will personalize its functionality for this new user.

On the one hand, this realization allows to implement specific mechanisms for personalization and conflict resolution in the service implementation, e.g. in case of the music service it is possible to search for some common music preference that all currently present users have in common. Alternatively, it is also possible to use priorities for each user or to keep playing music for the user who arrived first at the service's location. On the other hand, this realization requires to decide where to run the service in advance and individual service instances are needed for all locations. Furthermore, it requires to implement person management and conflict resolution mechanisms for each service again and again. This leads to a lot of implementation redundancy and contradicts to reuse. In addition, while every service implements its individual mechanisms, this can lead to non-uniform behavior.

Another pattern is a *person-bound realization* which means that the service instance is no longer bound to a fixed location but to a specific user. Now, the service instance "follows" the user. This means that the service is bound to the user's current location at any time and this association is changed according to the user's movement, thereby supporting in-home mobility. Since the service is now related to one specific user it only needs to access personal data for this user. In this realization person management and conflict resolution have to be handled by the middleware.

Yet, we have described, how we support in-home mobility by the different kinds of service realization. However, a person-bound realization is also the basis supporting inter-home mobility. Details of inter-home mobility support will be discussed in the following section.

### 3.3. Mobility Support

Considering inter-home mobility, we want to enable hassle-free access to visited environments, while allowing users to keep their preferences for personal services across multiple environments. Therefore the visited environment must have access to these preferences. There are several ways of how to provide personal data to visited environments.

One way would be to store the users' profiles in a central (Internet-based) repository. Every visited eHome would then get access to the profiles of its "logged-in" users. A major disadvantage of that solution is that all the personal data, including sensitive data such as medical data, is managed by the central repository and requires the users to trust this repository. However, many users may not be willing to do this. Another way would be to interconnect the visited eHomes and the user's "home" environment where the personal data is stored. The downside of this approach is that it requires to tell the visited eHomes where one is coming from. This makes anonymity of users difficult and conflicts with the protection of privacy. The third way would be to store personal data on a mobile device. Taking along personal data on a mobile device allows a user to release his preferences to visited eHomes on demand. This is depicted in Figure 3c.

We went for the last alternative as it does not have the disadvantages mentioned above. We refer to this approach as "client side personalization" [1]. A similar approach is also suggested in [4].

The user model is responsible for exchange of personal data between a user's mobile device and the visited environment, see again Figure 3c. This includes transfer of data to the environment during log-in, synchronization during the session, and deletion after log-out. For more details about the user modeling component, including privacy aspects, see [3].

Up to now, we assumed that functionalities which a user desires are realized by already running services in the visited environment. In this case, it would be sufficient only to transfer the necessary personal data to the environment. However, there might be situations, where the visited environment does not run the wished services. Now the question arises how a user still can be served with the desired functionalities.

We have extended our approach so that a user can take along also personal services, in addition to personal data, and execute them on his mobile device when needed. An example is shown in Figure 3d. On the residential gateway a proxy service is deployed which encapsulates the connection to the mobile device where the actual service is running. Usually, the service needs

to be bound to basic or integration services running in the eHome. Proxy services encapsulating the connection to these (remote) services, e.g. Speaker Control, will be generated on the mobile device and bound to the actual service on the mobile device, e.g. Music. This realization has the same effect from the user's point of view as if the mobile service would be running on the residential gateway.

Beside providing the user his desired functionality, this approach has another important advantage. Whenever a personal service is executed on a user's mobile device, the necessary personal data can be kept confidential on this mobile device. Thus, the amount of personal data transferred to the environment is reduced and the privacy protection enhanced. This is an important requirement, especially when moving through different and possibly unknown environments. However, this approach has also some disadvantages. It implies e.g. higher communication effort regarding service interaction between mobile device and the residential gateway. Furthermore, the energy consumption of the mobile device increases.

### 3.4. Configuration Support

As described in Section 2, top-level services are usually bound to further services, which can be integrating or basic services. There are several events which can affect a configuration.

One type of these events occurs when a device appears to a location. If the basic service controlling the new device is required by some other service bound to the same location, it will be bound to this service. In case of a disappearing device, it might happen that a location-bound service will be marked as invalid after its required service is unbound. These actions are similar for location-bound and person-bound services.

Other events occur when a person moves from a location to another one. In these cases, only personal services are affected. The case of location-bound personal services has been described previously in Section 3.2. In case of person-bound personal services, the situation gets more complex. Here, we have to distinguish two situations. If the service is running on the residential gateway, the personal service first will be paused. Next, the bindings of that service to basic services bound to the location which the user has left will be released. Then, the personal service will be bound to basic services providing the same functionality bound to the location which the user has entered. Finally, the service will be restarted. If the service is running on the mobile device, the proxy of the top-level service on the residential gateway is treated as the original service.

Important is, however, that new proxy services encapsulating communication to the required services in the new location have to be generated accordingly on the mobile device.

### 3.5. Implementation Details

We use the Java-based OSGi component model for eHome services. OSGi provides a SOA-based runtime environment for services and applications. The mobile implementation is based on top eRCP, an OSGi implementation for embedded systems. Unfortunately, OSGi and, thus, eRCP do not support distribution of services over multiple gateways. Due to this, we have realized remote communication between the mobile device and the eHome gateway via WLAN based on JXTA, a language-independent P2P protocol. We implemented our own RMI-like communication over JXTA, called "SimpleRMI", to enable distributed service interaction over multiple gateways [1].

Furthermore, we have extended our configuration approach to support dynamic and distributed service composition and deployment [7]. A light-weight version of the eHome Base component for mobile devices is used for this purpose. The personal data is modeled and exchanged based on the user model markup language USERML. The services interpret the data according to the general user modeling ontology GUMO [6].

We evaluated the mobile device software on Dell Axim X51v PDAs, capable of WLAN. As Java virtual machine for mobile devices we used IBM's WebSphere Everyplace Micro Environment. The evaluation of the gateway side was done in combination with our existing eHome prototype. This prototype contains a 2D simulation environment, the eHomeSimulator, which can be executed on usual computers to simulate different smart environments [2]. We have tested several services such as the Wake-up, Music, or Personal Room Temperature etc.

## 4. Related Work

*Mobile Gaia* [5] is a middleware which enables adhoc personal active spaces. A user can integrate his mobile devices to a personal active space for realizing certain functionality. However, the authors do not describe how Mobile Gaia can be used for connecting mobile devices with usual smart environments, called active spaces in Gaia terminology. Furthermore, Mobile Gaia assumes that each mobile device runs applications either of coordinator or client mode. In our approach



the a mobile device is used for different purposes. Besides running personal services, it can be also used to manage and release stored personal data.

The Aura project [8] aims at preserving continuity when a user moves between different environments. This is done by storing user task data on a global file server and by connecting the different environments to this server. In contrast to that, we let the users take along their data and even their services on a mobile device. This way, the user can control which parts of his personal data to release to a new environment.

Agents play an important role in the *MavHome Project* [9]. There are three main goals in this project: Maximizing living comfort, minimizing resource consumption, and maintaining safety and security of inhabitants. These goals are achieved by treating environments as intelligent agents. In contrast to that, we do not use agent technology but service composition based on an adaptive configuration process. Since we support different service realization patterns, we propose an approach based on a global view on the current state of an eHome system. Also, to our knowledge, inter-home mobility is not considered in the MavHome project.

Roduner et al. have analyzed the strengths and limits of using a mobile device as a universal interaction device in ubiquitous computing environments. They developed a system called AID for this purpose. Several tests have proved that persons using AID are faster solving exceptional tasks but slower solving every day tasks compared to executing these tasks on the appliances' own user interfaces. In our project we have also developed a prototype for mobile devices providing eHome users a unified user interface for interacting with personal and non-personal top-level services [1]. In contrast to AID, we additionally enable executing services and storing personal data on the mobile device for personalizing environments.

## 5. Summary and Outlook

In this paper we discussed different patterns of realizing eHome services. We evaluated these patterns and analyzed their applicability for mobility support and personalization. There is no single pattern that covers all scenarios that can occur in eHome systems. Therefore, we implemented a prototype which supports the execution of eHome services according to all discussed patterns. We could successfully show the applicability of our approach in a test environment consisting of a simulation environment [2] and several mobile devices.

We have also done some research on protecting the privacy of mobile users by use of anonymous credential

systems. However, we could not discuss the results in this paper, a respective publication is pending.

The presented patterns need to be evaluated in a representative study in order to assess their applicability in real world scenarios. Therefore, we are looking for industrial partners providing a large-scale testbed.

## References

- [1] I. Armac and D. Evers. Client Side Personalization of Smart Environments. In *SAM 2008: Proc. of the 1<sup>st</sup> Intl. Workshop on Software Architectures and Mobility at ICSE 2008*, pages 57–59. ACM, 2008.
- [2] I. Armac and D. Retkowitz. Simulation of Smart Environments. In *Proceedings of the IEEE Intl. Conf. on Pervasive Services 2007 (ICPS'07)*, pages 257–266. IEEE Press, 2007.
- [3] I. Armac and D. Rose. Privacy-Friendly User Modelling for Smart Environments. In *Proceedings of the The Fifth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2008)*. ACM, 2008.
- [4] L. Bass and J. Klein. Implications of a Single Mobile Computing Device. In *SAM '08: Proc. of the 1<sup>st</sup> Intl. Workshop on Software Architectures and Mobility*, pages 51–52. ACM, 2008.
- [5] S. Chetan, J. Al-Muhtadi, R. Campbell, and M. D. Mickunas. Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing. In *IEEE Consumer Communications and Networking Conference*, pages 223 – 228, Las Vegas, Nevada, January 2005. IEEE Computer Society.
- [6] D. Heckmann, T. Schwartz, B. Brandherm, M. Schmitz, and M. von Wilamowitz-Moellendorff. Gumo - the general user model ontology. In *Proceedings of the 10<sup>th</sup> International Conference on User Modeling*, pages 428–432, Edinburgh, UK, 2005. LNAI 3538: Springer, Berlin Heidelberg.
- [7] D. Retkowitz and M. Stegelmann. Dynamic Adaptability for Smart Environments. In R. Meier and S. Terzis, editors, *Distributed Applications and Interoperable Systems, 8<sup>th</sup> IFIP WG 6.1 International Conference (DAIS 2008)*, volume 5053 of *LNCS*, pages 154–167. Springer, 2008.
- [8] J. P. Sousa and D. Garlan. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *WICSA 3: Proc. of the IFIP 17<sup>th</sup> World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture*, pages 29–43. Kluwer, B.V., 2002.
- [9] G. M. Youngblood, L. B. Holder, and D. J. Cook. Managing Adaptive Versatile Environments. In *PERCOM '05: Proc. of the 3<sup>rd</sup> IEEE Intl. Conf. on Pervasive Computing and Communications*, pages 351–360. IEEE Computer Society, 2005.

# A Graph Transformation-Based Approach to Task Allocation in Wireless Sensor Actor Networks

Hossein Momeni  
Computer Engineering  
Department  
Iran University of Science  
and Technology  
momeni@iust.ac.ir

Vahid Rafe  
Computer Engineering  
Department  
Arak University  
Arak, Iran  
rafe@iust.ac.ir

Mohsen Sharifi  
Computer Engineering  
Department  
Iran University of Science  
and Technology  
msharifi@iust.ac.ir

Adel T. Rahmani  
Computer Engineering  
Department  
Iran University of Science  
and Technology  
rahmani@iust.ac.ir

## Abstract

Existing WSNs suffer from the lack of a real-time task allocation in support of real-time communication and coordination. In this paper we present a graph transformation-based approach to allocate the tasks to the sensor and actor nodes in support of real-time application. For each action in the designed scenarios we define one or more graph rules to implement the proposed model. Using this formalism we analyze the correctness of our algorithms. We show that the proposed approach guarantee that the tasks complete their activities before their deadlines expire. To show the efficiency of our approach we have simulated the model. Simulation results showed an improvement of 65 percent in deadline hit ratio comparing our approach to FIFO algorithm.

## 1. Introduction

Wireless sensor actor networks (WSANs) consist of a group of sensor and actor nodes that perform distributed sensing and acting tasks [1]. Sensors are able to sense environmental information and actors are able to act upon environment. These nodes communicate wirelessly and dynamically to monitor and control environment. Figure 1 shows a typical architecture of WSNs. In this architecture, the task manager node sends its request through Internet and satellite to the sink and the sink as a central computer dispatches these requests to designated sensor and actor nodes that in turn send back the results to the sink.

The existence of actors in an environment is demonstrative of time in performing some tasks; hence real-timeliness is an important issue in WSNs [1] to guarantee that tasks of an application are distributed based on limited computation and communication

resources in the network in such a way to meet their deadline.

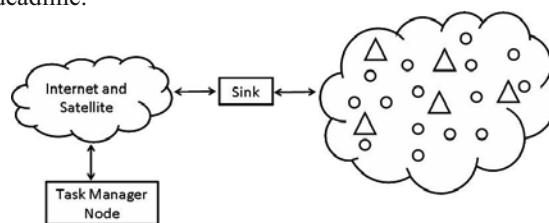


Figure1. A typical architecture of WSNs

In spite of the importance of real-time requirements in WSNs, this issue had not been totally resolved yet. Existing researches, e.g., [2], have tried to minimize delays in packet transmission, in real-time routing and in coordination, but fall short of guaranteeing task completion time before specified deadlines expire. To provide such guarantees, tasks must be allocated to network nodes considering the deadlines. This had been pursued only in wireless sensor networks (WSNs).

In this paper we present a new two level task allocation mechanism for WSNs by allocating tasks to sensor nodes and actor nodes considering the real time deadlines of tasks. Our mechanism considers two tasks: sensing and acting tasks. Candidate sensor nodes and actor nodes that have enough energy and can perform the tasks are identified first, and then tasks are dispatched to them based on deadlines.

To do so, we first model formally the required features of WSNs by graph transformation systems. Graph transformation has recently become more and more popular as a general formal modeling language [3]. We use this model to analyze the soundness of our algorithms. Furthermore, using the AGG toolset [4], we analyze the modeled WSNs and check the results to become sure about the correctness of the model. Finally, the efficiency of our approach is evaluated by simulation. We simulated our proposed algorithms in VisualSense [7].

The rest of this paper is organized as follows. Section 2 explains some terminologies and assumptions. Section 3 describes the model of system using graph transformation systems. Section 4 presents our approach. Section 5 shows the experimental results and Section 6 concludes the paper.

## 2. Assumptions and Terminologies

In this section we state our assumptions for task allocation in WSANs, besides a brief description of graph transformation systems.

Formally, we define a task as follows:

$$T_i = \langle \text{period}_i, \text{deadline}_i \rangle, 1 \leq i \leq N$$

Where:

- $\text{period}_i$ , is the period of  $T_i$  meaning that  $T_i$  is executed once every  $\text{period}_i$  units of time. The deadline for  $T_i$  is equal to  $\text{deadline}_i$ , i.e. the time that execution of  $T_i$  must be completed.

The execution of the first task will be started with the period of the task, we define two tasks: sensing and acting.

For remote task communications we use a tuple space communication architecture originally proposed in Linda [5]. Tuples are collections of passive data values. A tuple space is a pool of shared information, where tuples can be inserted, removed or read [6].

We introduce two tuple spaces that share information about sensors and actors among cluster heads and the sink respectively. Cluster heads and the sink can use the tuple space to retrieve information about the status of sensor nodes such as energy of each node. Figure 2 shows tuple space architecture for WSANs.

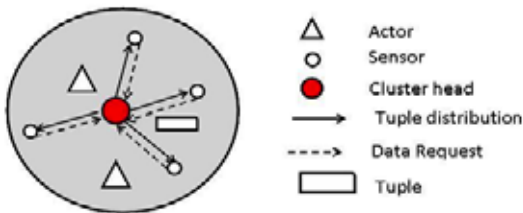


Figure 2. Tuple space architecture for WSANs

Tuples are defined as follows:

- If  $S = \{SN_1, SN_2, \dots, SN_L\}, |SN| = L$  is a set of L sensor nodes and  
 $Sense = \{Sense_1, Sense_2, \dots, Sense_M\}, |Sense| = M$

is a set of M sensing tasks, then the first tuple space is defined as follows:

$$\Delta_{j,k} = \langle \text{sense}_{j,k}, \text{delay}_{j,k}, \text{power}_{j,k} \rangle$$

where  $1 \leq k \leq M, 1 \leq j \leq l$  and  $\Delta_{j,k}$  describes that the  $j$ th sensor performs the  $k$ th sensing tasks with dilation time  $\text{delay}_{j,k}$  and power consumption  $\text{power}_{j,k}$ .

- If  $A = \{A_1, A_2, \dots, A_Z\}, |A| = Z$  is a set of Z actor nodes and

$$\text{Action} = \{ \text{Action}_1, \text{Action}_2, \dots, \text{Action}_X \}$$

where  $|\text{Action} = X|$  is a set of X acting tasks, then the second tuple space is denoted by:

$$\psi_{j,k} = \langle \text{Action}_{j,k}, \text{delay}_{j,k}, \text{power}_{j,k} \rangle$$

where  $1 \leq k \leq X$  and  $\psi_{j,k}$  describes the  $j$ th actor that performs the  $k$ th acting task with dilation time  $\text{delay}_{j,k}$  and power consumption  $\text{power}_{j,k}$ .

## 3. Graph Transformation Model

The mathematical foundation of graph transformation systems returns to thirty years ago in reaction to shortcomings in the expressiveness of classical approaches to rewriting (e.g. Chomsky grammars) to deal with non-linear grammars. In this subsection, we describe graph transformation briefly, as a modeling means. For more information about theoretical background and semantics of graph transformation, interested readers can refer to [8,9].

**Definition 1 (attributed type graph transformation).** An attributed type graph transformation system is a triple  $AGT = (TG, HG, R)$ , where  $TG$  is the type graph,  $HG$  is the host graph and  $R$  is the set of rules.

**Definition 2 (Type Graph).** Let  $TG_N$  be a set of node types and  $TG_E$  be a set of edge types. Then a type graph  $TG$  is a tuple:  $TG = (TG_N, TG_E, src, trg)$ , with two functions  $src: TG_E \rightarrow TG_N$  and  $trg: TG_E \rightarrow TG_N$  that assign to each edge a source and a target node.

**Definition 3 (Host Graph).** A host graph  $HG$ , also called *instance graph* over  $TG$ , is a graph equipped with a graph morphism  $type_G: HG \rightarrow TG$  that assigns a type to every node and edge in  $HG$ .

**Definition 4 (Graph Rules).** In this paper, we follow the algebraic double pushout approach (DPO) to graph transformation as first introduced by Ehrig et als. for untyped graphs in [9]. A graph transformation rule  $P$

over an attributed type graph  $TG$  is given by  $P = (L \xleftarrow{l} K \xrightarrow{r} R, \text{type}, \text{NAC})$ , where:

- $L \xleftarrow{l} K \xrightarrow{r} R$  is a rule span with injective graph morphisms  $l, r$  and graphs  $L$  (Left Hand Side or LHS),  $K$  (gluing graph) and  $R$  (Right Hand Side or RHS) typed over  $TG$ .
- $\text{type} = (\text{type}_L: L \rightarrow TG, \text{type}_K: K \rightarrow TG, \text{type}_R: R \rightarrow TG)$  is a triple of morphisms, and
- $\text{NAC}$  is a set of triples  $\text{nac} = (N, n, \text{type}_N)$  with  $N$  being a graph,  $n: L \rightarrow N$  a graph morphism, and  $\text{type}_N: N \rightarrow TG$  a morphism.

The application of a rule to a host graph  $H$ , replaces a matching of the LHS in  $H$  by an image of the RHS. This is performed by (1) finding a matching of LHS in  $H$ , (2) checking the negative application condition  $\text{NAC}$  (which prohibits the presence of certain nodes and edges) (3) removing a part of the host graph (that can be mapped to LHS but not to RHS) yielding the context model, and (4) gluing the context model with an image of the RHS together by adding new nodes and edges (that can be mapped to the RHS but not to the LHS) and obtaining the derived model  $H'$ .

#### 4. Proposed Approach

In this section, we describe our approach to model the WSANs using graph transformation systems.

As it was mentioned, in a graph transformation system, the type graph shows the metamodel of the system. Hence, designing the type graph is the first step to model the system.

Figure 3 shows the proposed type graph as the metamodel of the system. It comprises 10 nodes with different types. For example, node “clock” shows the current time in the model. It has an attribute names “Time” to show the current time. We use this node to simulate the time. The behaviors (rules) which must be done at the same time on the model do not change the time but other rules change it. The node “Sensor” is an abstract node, the two nodes “ClusterHead” and “TypicalSensor” have inherited it. In fact, each sensor in the model, ether is a cluster head or is typical sensor. Also, there is an association edge between “ClusterHead” and “TupleSpace”. It means each cluster head in the model has a tuple space witch contains triples called “Tuple”. Each “TypicalSensor” stores its current status as a “Tuple” in the “TupleSpace”. We design this type graph based on the assumptions in section 2. For example the node “Tuple” contains all the attributes which we defined them in the previous section.

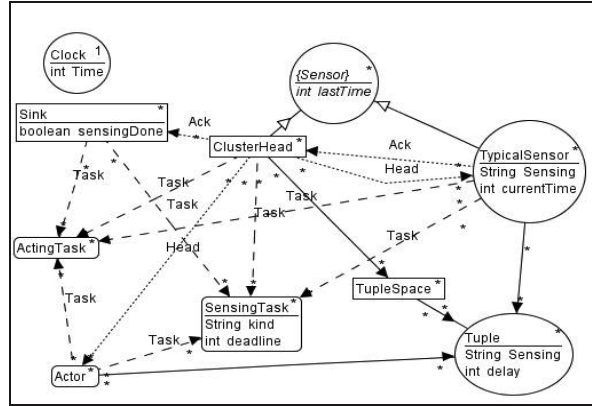


Figure 3. The proposed type graph for WSANs model

This type graph shows the proposed syntax for the models. According to this type graph, we can model the initial configuration of the WSAN as a host graph. For example, figure 4 shows a small WSAN with 3 sensors, 1 cluster head, 1 sink, 1 tuple space and so forth. In general, the host graph is served as the starting point for simulation and analysis.

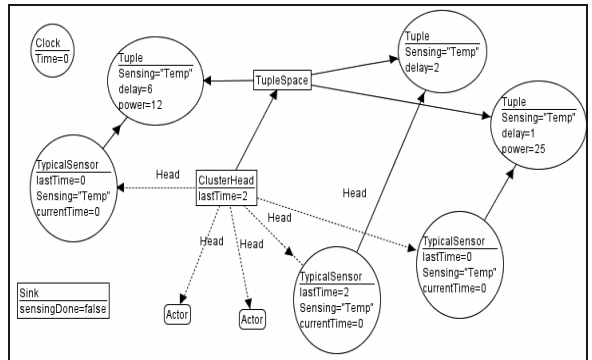


Figure 4. A host graph to show the initial configuration of a small WSAN

We define graph rules to implement the desired semantics (based on the proposed approach). For each behavior which must be done on the model, one or more graph rule(s) is defined. For example, consider rule of figure 5. It shows the creation of sensing task. The left side of this rule says if there is a sink in the model which its “SensingDone” attribute is false and there is not a “SensingTask” associated to it (negative application condition), then a sensing task will be generated by this rule. For this rule, we have supposed the kind of the task is sensing the temperature and its deadline is 4. For other cases (i.e. sensing tasks with different kinds and deadlines) we define similar rules.

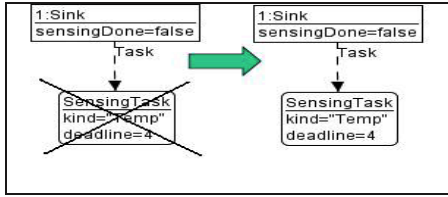


Figure 5. Rule of creating sensing task

As another example, consider rule of Figure 6. It shows the process of assigning sensing task to cluster heads by the sink node. The LHS and NACs together say that if the sink has a task and this task has not been assigned to a cluster head while both nodes have the same kind, then the task must be assigned to the cluster head.

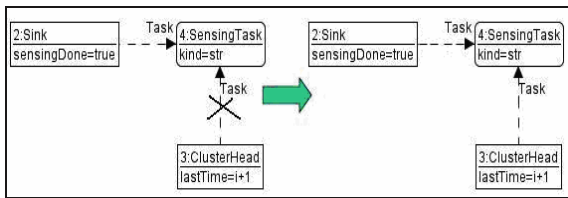


Figure 6. Rule of assign sensing task

## 5. Experimental Results

The effectiveness of our approach is validated through simulation. We simulated our proposed approach in *VisualSense* [7].

We first model sink, sensors and actors so that each of these nodes has *TypedCompositeActor* that include time component, send message component and receive message component. Also each *TypedCompositeActor* has *TypedAtomicActor* that is processor of each node. In the experiments, we counted the number of tasks that is rejected by nodes and the number of deadline that is missed. We compared our approach with FIFO mechanism and we observed that our approach provides the least rejected tasks and reduce deadline miss.

The results shown in Figure 7 indicate that the deadline misses in the proposed approach is low compared to FIFO algorithm.

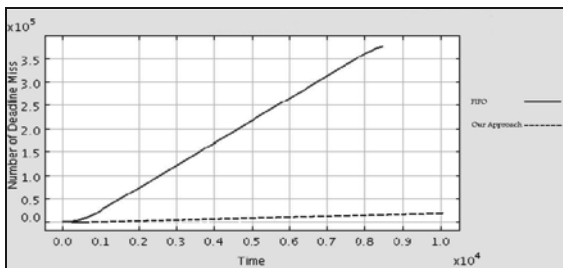


Figure 7. Deadline misses

## 6. Conclusion

In this paper we proposed A Graph Transformation-Based Approach to Task Allocation to prevent missing the deadlines of such tasks. For this purpose, before allocating tasks to nodes, we checked the feasibility of execution of tasks on those nodes based on their power and deadline requirements. We modeled formally the required features of WSANs by graph transformation systems. We used this model to analyze the soundness of our approach. We analyzed the modeled WSANs and checked the results to being sure about the correctness of the model. Finally, the efficiency of our approach was evaluated by simulation. Experimental results showed an improvement of 65 percent in deadline hit ratio compared to FIFO algorithm.

## 7. References

- [1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges", Ad Hoc Networks, vol. 2, 2004, pp. 351-367
- [2] G. A. Shah, M. Bozyigit, Ö.B. Akan and B. Baykal, "Real-Time Coordination and Routing in Wireless Sensor and Actor Networks", Lecture Notes in Computer Science (LNCS), No. 4003, Springer-Verlag, 2006, pp. 365-383
- [3] L. Baresi, V. Rafe, A. T. Rahmani and P. Spoletini, "An Efficient Solution for Model Checking Graph Transformation Systems", Electronic Notes in Theoretical Computer Science (ENTCS), vol. 213, 2008, pp. 3-21.
- [4] AGG, [tfs.cs.tu-berlin.de/agg/](http://tfs.cs.tu-berlin.de/agg/).
- [5] D. Gelernter, "Generative communication in linda," ACM Trans. Programming Languages and Systems, vol. 7, no. 1, 1985, pp. 80-112.
- [6] M. Koorilehto, M. Hännikäinen and T. D. Hämmäläinen, "A Survey of Application Distribution in Wireless Sensor Networks", EURASIP Journal on Wireless Communications and Networking, vol. 5, no. 5, 2005, pp. 774 - 788.
- [7] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, Y. Zhao, C. Brooks, N. V. Krishnan, S. Neuendorffer, C. Zhong, and R. Zhou, Visualsense, "Visual Modeling for Wireless and Sensor Network Systems", Electronics Research Laboratory, College of Engineering, University of California, 2004.
- [8] L. Baresi, R. Heckel, "Tutorial Introduction to Graph Transformation: A Software Engineering Perspective", First International Conference on Graph Transformation, LNCS, vol. 2505, 2002, pp. 402-429.
- [9] H. Ehrig, G. Engels, H. Kreowski, G. Rozenberg, "Handbook on Graph Grammars and Computing by Graph Transformation, Applications", Languages and Tools, vol. 2, World Scientific. 1999.

# Another New Criterion to Improve the Interaction Diagrams Quality

Lilia Grati, Mohamed Tmar and Faïez Gargouri

*MIRACL Laboratory, Higher institute of informatics and multimedia of Sfax,*

*BP 242 - 3021, Sakiat Ezzit, Sfax, Tunisia.*

*{lilia.grati, mohamed.tmar}@isimsf.rnu.tn, faiez.gargouri@fsegs.rnu.tn*

## Abstract

*UML interaction diagrams are used to represent the dynamic relationships that exist among the system's objects. These links are established through exchanging the messages between the objects, which is considered as a coupling aspect. Actually, this concept corresponds to the inter-dependency between the diagram's objects and, as it was confirmed by several researchers, the minimization of coupling is considered as a quality indicator.*

*In this paper, we lead an experimental analysis study and propose a new criterion to estimate and to improve the quality of the interaction diagrams and especially, to control the communication and the representation of messages in these UML diagrams.*

## 1. Introduction

UML interaction diagrams show the dynamic aspect of the information system in terms of the exchanged messages between the objects. Indeed, the main purpose of these diagrams is to model the use-cases scenarios reflecting the users' requirements. For that, it will be useful to propose different criteria to improve their quality's level. In fact, a great number of measures and heuristics were suggested to be applied in these diagrams. Nevertheless, most of them lack valid and clear definitions. Therefore, it is difficult for designers to make a sensible choice.

Thus, this paper is consecrated to present, firstly an overview on two criteria that we proposed in a previous work. The second part of the paper is focused on presenting a new criterion that aims to analyze the impact of the communication manner between the objects of an interaction diagram, on its quality.

## 2. Proposition of our quality criteria

The main objective of our works is to propose different criteria that can evaluate and improve the interaction diagrams quality. Generally, designers

consider that an interaction diagram has a "good" quality if it corresponds to the use-case's scenarios.

However, if we refer to the different measures and heuristics that exist in the literature, such as [1], we presume that the researchers confirm that minimizing coupling between classes or objects is considered as a quality indicator.

In order to determine the adequate values corresponding to the objects and messages numbers in an interaction diagram, we have led an empirical study and statistical calculations based on two corpuses of sequence diagrams containing respectively 66 "good" diagrams and 127 "bad" ones.

In fact, a sequence diagram quality can be considered as "bad" if the diagram does not reflect the corresponding use-case scenario, or if it contains errors of syntax and form, or if the representation of the exchanged messages and the intervening objects of the considered diagram, needs to be modified and optimized.

We denote that we have selected the "good" sequence diagrams of our first corpus from the following books: [2], [3], [4] and [5].

Our method aims to lead statistical calculations in order to show the variations of each criterion, applied in the two corpuses of sequence diagrams, by curves.

Any difference that can be observed through the comparison of the curves corresponding to every criterion is an indicator of its relevance. In other words, if the curves, resulting from the application of a criterion in the "good" and the "bad" sequence diagrams, are similar then, we deduce that this criterion have no impact on the quality level of the interaction diagrams.

### 2.1. Overview on our two previous criteria

In [6], we have proposed two quality criteria, referring to the minimization of coupling through the verification of the exchanged messages between objects and also, the analysis of the distribution of the messages in an interaction diagram.

**2.1.1. Verification of the exchanged messages between objects.** The minimization of the exchanged messages between the interaction diagram's objects is considered as a quality criterion in this kind of UML diagrams as it reduces coupling.

Thus, we have proposed in [6], to analyze and to test the variation of the number of messages in each category of the diagrams that we have previously modeled.

However, reducing only the messages exchanges between the objects is insufficient neither to evaluate nor to improve the diagram's quality. For example, an interaction diagram containing an only one message can not be considered as an "excellent" diagram. Besides, a diagram that contains a great number of objects but a very little number of messages will be certainly classified as a "bad" one.

Accordingly, we assume that the number of messages is correlated to the number of objects belonging to an interaction diagram. This dependency is taken into account through determining and justifying the adequate value of the number of messages in diagrams having a well-known value of objects number. So, we aimed to experiment this correlation between objects and messages belonging to an interaction diagram, based on this formula:

$$x = \frac{Nm}{No} \quad (1)$$

' $x$ ': the relative number of messages in the interaction diagram ' $D$ ';

' $Nm$ ': number of messages in ' $D$ ' diagram.

' $No$ ': number of objects in ' $D$ ' diagram.

Based on the calculations that were applied on the 66 "good" diagrams and the 127 "bad" ones, we have obtained two different curves, drawing respectively the variation of the number of the "good" and the "bad" diagrams depending on the relative number of messages: ' $x$ '. Thus, we deduced that a difference exists between the values of the relative number of messages: ' $x$ ', to which corresponds the maximum of each curve. In fact, this difference implies the possibility to consider the relative number of messages that we have defined, as a relevant quality criterion in the interaction diagrams which contributes to judge and so, to improve the quality of the representation of the messages in this type of UML diagrams.

This assumption was demonstrated and approved through all the calculations that were detailed in [6].

**2.1.2. Verification of the activity and the reception degrees of the interaction diagrams.** Our second quality criterion that we proposed in [6], aimed to study the distribution of the messages between the interaction diagram objects.

In fact, a "good" interaction diagram should not have very active objects and passive ones. A very

active object sends a lot of messages to the other objects. Moreover, if an object sends (or receives) a great number of messages to the same object of an interaction diagram, it is probable that these objects have a strong relationship and describe the same abstraction and so, the designer should have a feedback and for example, may group them into a same class. Besides, every object that belongs to a "good" interaction diagram must be active by sending an adequate number of messages.

The global activity degree:  $Act(D)$  of an interaction diagram ' $D$ ', having ' $No$ ' objects and ' $Nm$ ' messages, is measured by the product of all the elementary activities of its objects. We define this measure as follows:

$$Act(D) = \prod_{k=1}^{No} act(O_k) \quad (2)$$

$act(O_k)$  represents the elementary activity of an object ' $O_k$ ' belonging to the interaction diagram ' $D$ '. It is defined by formula 3:

$$act(O) = \frac{nm_s + 1}{Nm + 1} \quad (3)$$

' $nm_s$ ': number of messages that are sent by the object ' $O$ ' of the diagram ' $D$ '.

As a result from using the product to calculate the global activity degree of an interaction diagram ' $D$ ', the value of this activity can be maximal if and only if all the objects of ' $D$ ' send approximately the same number of messages.

The results of our experimental study that we have applied to the two corpuses of sequence diagrams are shown by two different curves drawing respectively the variation of the global activity degrees in the "good" and the "bad" diagrams. We have observed that the global activity degrees in "good" diagrams are clearly higher than the "bad" diagrams ones. Consequently, we deduced that the objects belonging to the "good" diagrams send almost the same number of messages and so, there are no passive objects in these diagrams.

Since we aim to control the distribution of the messages in the interaction diagrams, our interest was also focused, in [6], on studying the reception of messages by the objects of a diagram. So, we defined the global reception degree of an interaction diagram ' $D$ ', by the product of the elementary reception degrees of all the objects belonging to this diagram:

$$Rec(D) = \prod_{k=1}^{No} rec(O_k) \quad (4)$$

$Rec(D)$ : the global reception degree of an interaction diagram ' $D$ ;

$rec(O_k)$ : the elementary degree of reception of messages by the object ' $O_k$ ' in the diagram ' $D$ '.

This criterion is measured by formula 5:

$$rec(O) = \frac{nm_R + 1}{Nm + 1} \quad (5)$$

' $nm_R$ ': number of messages that are received by the object ' $O_k$ ' belonging to the diagram ' $D$ '.

As a result from the application of this criterion in the two corpuses of diagrams, we have obtained two different curves corresponding respectively to the variations of the global receptions of messages, in the "good" and the "bad" diagrams: the global reception degrees in the "good" diagrams are clearly higher than the "bad" diagrams ones. Accordingly, we presume that all the objects belonging to a "good" interaction diagram receive approximately the same number of messages. So, the reception of the messages in the first corpus of sequence diagrams is well shared out between the objects of these "good" diagrams contrary to the "bad" ones.

We note that the details of all the calculations and also of all the curves corresponding to the measures are presented in [6].

## 2.2. Our new criterion: verification of the dependency degree between the objects of an interaction diagram

We still aim to improve the interaction diagram quality through the minimization of coupling and for that, we propose, in this subsection, another new criterion. It relies on studying the communication and the interdependency between the couples of objects.

In a "good" interaction diagram, the communication between the pairs of objects must be decentralized and the exchanged messages between them must be well distributed. So, we should not find an only one couple of objects that exchanges the majority of the messages of the diagram since, it is probable that one of these two communicating objects belongs to an abstraction level that is higher than all the other objects of the diagram.

For example, if the designer uses the object: "System" in an interaction diagram, this object will certainly monopolize the behavior of the diagram as it is considered as a "generic" object representing all the objects of the modeled information system.

We present, in the following, another example to explain how can the bad distribution of messages and the high dependency between the objects, be an indicator on the "bad" quality of the diagram: if the designer uses the object: "Person" to model scenarios of the "commercial management" domain then, the majority of the messages will certainly be sent and received by this object as it is general and represents four objects at the same time: "Customer", "Supplier", "Administrator" and "User".

Accordingly, we assume that the high dependency between the objects of an interaction diagram is an indicator of its quality level and so, the designer

should make a feedback and check the objects of its modeled diagram. In order to study this criterion and to test its impact on the "good" and the "bad" diagrams quality level, we propose firstly, to calculate the dependency degree " $DD$ " between the communicating objects as follows:

Let ' $O_i$ ' and ' $O_j$ ' be two communicating objects belonging to the interaction diagram: ' $D$ ':

$$DD(O_i, O_j) = \frac{\text{Number of messages exchanged between } O_i \text{ and } O_j}{\text{Total number of messages sent or received by every object: } O_i \text{ and } O_j} \quad (6)$$

In fact, having a high degree of dependency between two objects indicates that they exchange a big number of messages and they have a great correlation. So, the representation of messages in the considered diagram needs to be optimized.

As we intend to valid theoretically our proposition that is focused on studying the impact of the centralized communication between objects on the quality of the diagram, we are based on the principles of an automatic method of classification: "*the hierarchical classification*". This method is a branch of the data analysis that aims to product groups of elements to form classes. The principle of the hierarchical classification algorithm is to gather successively the nearest elements having the smallest distance, into classes [7], as it is shown by figure 1. For this reason, it is indispensable to define and propose a distance to apply the hierarchical classification in our study case.

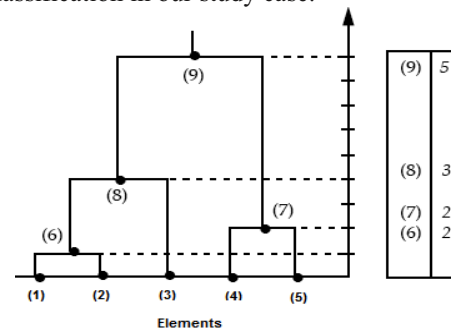


Figure 1. Hierarchical tree [7]

In our context, two objects are gathered in a partition (or class), if and only if they exchange a great number of messages. This aspect will be measured by a well defined distance that separates between two objects. Having a very small distance between two objects gathered into the same class, indicates that they are too interdependent and correlated and thus, their dependency degree is very high. Consequently, the coupling between these objects is great, which implies that the distribution of the messages in the considered interaction diagram needs to be optimized.



Let ‘ $O_i$ ’ and ‘ $O_j$ ’ be two communicating objects belonging to an interaction diagram and having a dependency degree  $DD(O_i, O_j)$ . We define the **distance** separating these two objects as follows:

$$d(O_i, O_j) = \frac{1}{DD(O_i, O_j)} \quad (7)$$

After grouping the objects of an interaction diagram into classes and as we aim to form groups of classes, we have to calculate the **distance between two classes**, which is the average distance between two objects selected at random from these two classes. So, we define this distance as follows:

$$d(C_k, C_m) = \frac{\sum_{(O_k, O_m) \in C_k \times C_m} d(O_k, O_m)}{|C_k| \times |C_m|} \quad (8)$$

$|C_k|$ : is the number of objects in the class: ‘ $C_k$ ’.

After that, we propose to determine **the inertia** of each class in order to estimate the homogeneity degree existing between all the elements that are gathered into the class. This concept is calculated by formula 9.

$$Inertia(C) = \frac{\sum_{(O_i, O_j) \in C \times C} d(O_i, O_j)}{|C|^2} \quad (9)$$

$Inertia(C)$  corresponds to the average distance between any two objects belonging to ‘ $C$ ’ class.

We note that  $|C|^2$  refers to the number of comparisons established between the objects of the class for a considered assemblage.

The last step of our method relies on determining the **total inertia** of every sequence diagram and for each assemblage of objects into classes (groups). As a result of this step, we obtain all the curves corresponding to the total inertias of the “good” and the “bad” sequence diagrams.

The total inertia of an interaction diagram ‘ $D$ ’ for an assemblage of objects: ‘ $A$ ’ is the weighted average of the inertias of the classes belonging to ‘ $D$ ’ and formed in every assemblage ‘ $A$ ’ of objects.

The weighting coefficient is:  $\frac{|C|}{No}$

$|C|$  number of objects, gathered into the class ‘ $C$ ’;

‘ $No$ ’: number of objects in the diagram ‘ $D$ ’.

So, this coefficient aims to show the contribution of each class in the total inertia calculated for a considered interaction diagram. Thus, the more the class contains object, the greater its participation, in the total inertia of the diagram, is.

$$Total-Inertia(D, A) = \sum_{C \in (D, A)} \frac{|C|}{No} * Inertia(C) \quad (10)$$

In the case of the “good” interaction diagrams, the evolution of the total inertias curves is stable, on the

contrary of the “bad” diagrams ones showing a non stable, acute and strong evolution. This is in fact, a sign of a “bad” communication between the objects that were grouped into classes for the diagram.

Example: let us consider two different distributions of points, presented in figure 2.

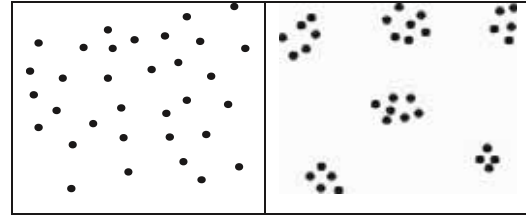


Figure 2. Example of two scatters plots

Based on calculations of distances between points, we aim to estimate the rapprochement and homogeneity degrees between points.

In the first scatter plot, it is observable that the elements that will be grouped into classes are well distributed in space. The distances separating each pair of points are almost similar. So, these elements are homogeneous and consequently, the curve drawing the total inertia of this first scatter plot will be approximately linear with a stable evolution (see curve 1 of figure 3).

In the case of the second scatter plot, it is clear that the elements can be grouped in six distinct and disjointed classes. In each class, the elements are too homogenous since the distances separating every pair of them, are too low. As a result, the total inertia curve, corresponding to this second distribution of points, stills in the beginning stable. If we have to group two points belonging to two different classes, the distance that separates them will be certainly high and consequently, the evolution of the total inertia curve will change suddenly and will be strong and acute (see assemblage 8 of the second curve presented by figure 3).

Figure 3 shows an example of two different curves drawing the total inertia of each scatter plot, presented previously by figure 2. The horizontal axis represents the number of points’ assemblages to form classes. The vertical one corresponds to the total inertia of every distribution of points.

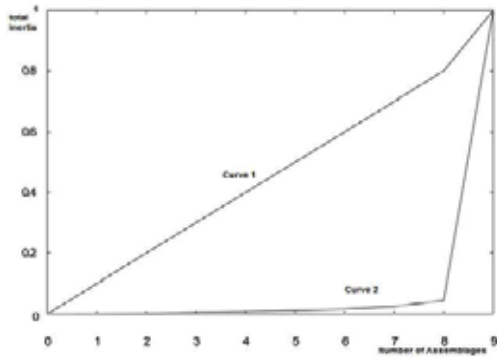


Figure 3. Example of two total inertia curves

As it is shown by figure 3, the second curve has, in a first phase, a stable and low evolution. But after the assemblage number 8, this evolution changes suddenly and becomes strong and acute. This is due in fact, to an assemblage between elements belonging to very disjoint classes. On the contrary, the first curve is approximately linear with a stable evolution.

❖ **Experiments and results:**

In order to experiment the impact of the high dependency degree between the objects of an interaction diagram, on his quality level, we started by modeling two corpuses of sequence diagrams containing respectively 66 “good” diagrams and 127 “bad” ones. Then, we followed these steps:

1. For each sequence diagram, we have calculated the number of messages that are exchanged between each pair of communicating objects. So, the pairs of objects that don't exchange messages are discarded from our calculations.
2. Then, we have applied successively formulas 6 and 7, to estimate the distances separating each pair of communicating objects of all sequence diagrams.
3. After grouping the nearest objects of each sequence diagram into classes, we have applied consecutively formulas 8, 9 and 10.
4. Finally, by averaging the total inertia evolution as function of the assembling iterations of the whole sequence diagrams, we have obtained the results that are presented by figures 4 and 5.

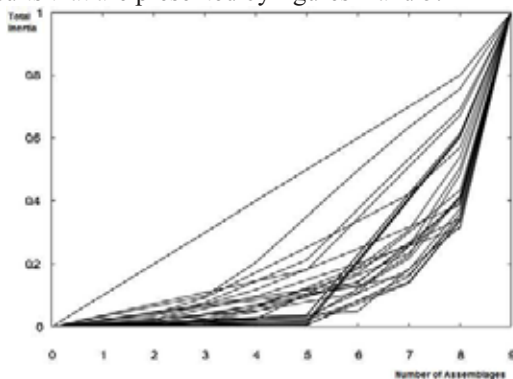


Figure 4. Total inertia curves of the “good” diagrams

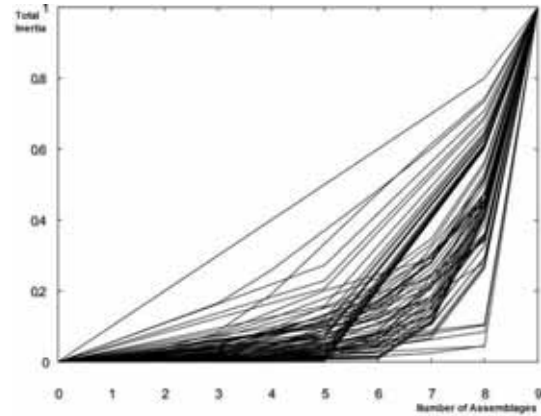


Figure 5. Total inertia curves of the “bad” diagrams

If we refer to figures 4 and 5, we observe that the total inertia curves corresponding to the “good” diagrams have a stable evolution, on the contrary of the “bad” ones having strong and acute evolution.

Generally, the total inertia curves that were presented by figures 4 and 5 have an exponential function:

$$f(x) = \exp(\alpha x + \beta) \quad (11)$$

‘x’ represents the number of assemblages that are established to form classes in an interaction diagram.

Let have:  $Y = \log(f(x))$

Thus, we obtain the following equation:

$$Y = \alpha x + \beta \quad (12)$$

Accordingly, we have to determine the values of the coefficients: ‘α’ and ‘β’.

Based on the linear regression method, we can calculate the value of the coefficient ‘α’ as follows:

$$\alpha = \frac{\sum_{x,Y} (x - \bar{x}) * (Y - \bar{Y})}{\sum_x (x - \bar{x})^2} \quad (13)$$

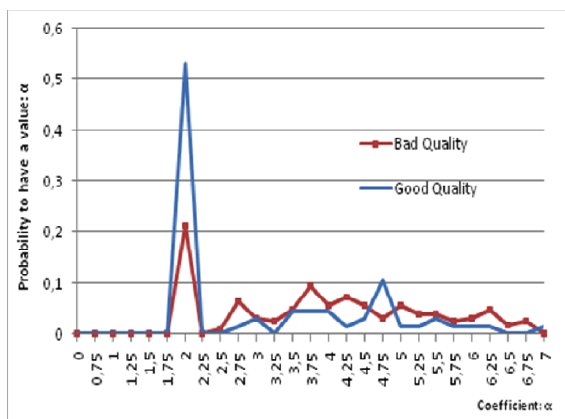
The evolution of a total inertia curve is measured by the next formula:

$$f'(x) = \alpha \exp(\alpha x + \beta) \quad (14)$$

Based on formula 14, the evolution of a total inertia curve depends principally on the coefficient ‘α’. If the value of the latter is high, the evolution of the curve will be less stable and more and more strong. This assumption will be proved experimentally through the comparison of the variation of ‘α’ between the “good” and the “bad” sequence diagrams.

To reach our objective, we have determined all the values of the coefficient ‘α’ corresponding to the total inertia curves of the “good” and the “bad” diagrams, by applying formula 13. Afterward, we calculated the probabilities to have respectively “good” and “bad” inertia curves depending on the coefficient ‘α’. The results of our experimentation are

shown by figure 6. The horizontal axis contains the values of the coefficient 'α'. The vertical one represents the probability to have a "good" or a "bad" sequence diagram as function of 'α'.



**Figure 6. Variation of the coefficient 'α' in the "good" and the "bad" inertia curves**

Figure 6 affirms experimentally our theoretical assumptions corresponding to the difference between the evolutions of the total inertia curves of the "good" and the "bad" sequence diagrams. If the value of the coefficient 'α' is high, the probability to have a "bad" total inertia curve and then, a "bad" sequence diagram is more and more great.

In fact, it is clear that starting from the threshold:  $\alpha = 2,25$ , it will be more probable to have a "bad" total inertia curve since the second curve, drawn after this value of 'α', is constantly on top of the first one that corresponds to the variation of 'α' in the inertia curves of the "good" sequence diagrams. This observation proves experimentally, our previous hypothesis and indicates that the evolutions of the "bad" total inertia curves are actually greater and acuter than the "good" ones and this is a sign of a bad communication between the pairs of objects in an interaction diagram. Accordingly, we affirm the relevance of our new proposed criterion.

### 3. Conclusion

This paper aims to present several quality criteria that can be applied in the interaction diagrams context in order to evaluate and then, to improve their quality level. In fact, based on the heuristics and the measures that exist in the literature, we are convinced that the minimization of the coupling aspect in an interaction diagram can improve the quality of its messages' representation. To reach our objective and to prove experimentally this problematic, we have led an empirical study and modeled two corpuses containing 193 sequence

diagrams having respectively "good" and "bad" quality levels.

We have presented, in first, two quality criteria that refer respectively to the verification of the exchanged messages between objects and the determination of the total activity and reception degrees in the interaction diagrams.

Our new criterion corresponds to the estimation of the dependency degree between the interaction diagrams objects, based on distances calculations. We have experimentally proved that if the majority of the messages of an interaction diagram, are exchanged between an only one pair of objects then, it is very probable that this diagram needs to be optimized because of its "bad" quality.

Our future works will be focused on proposing other criteria to improve the quality of the other diagrams provided by UML language.

### 4. References

- [1] Briand, L.C., Daly, J.W., and Wüst, J.K. A unified framework for coupling measurement in object-oriented systems. *IEEE transactions on software engineering*, 1999, vol. 25, N° 1, pp.91 – 121.
- [2] André, P., Valley, A. *Génie logiciel*. Editions Ellipses, 2003.
- [3] Kettani, N., Mignet, D., Page, P., Rosenthal-Sabroux, C. *De Merise à UML*. Editions Eyrolles, 2<sup>nd</sup> edition, 1999.
- [4] Roques, P. *UML par la pratique*. Editions Eyrolles, 2001.
- [5] Roques, P. *Les cahiers du programmeur*. Editions Eyrolles, 2<sup>nd</sup> edition, 2003.
- [6] Grati, L., Tmar, M., and Gargouri, F. Minimizing coupling to improve the interaction diagrams quality. In *Proceedings of the 15<sup>th</sup> conference on Information and Software Technologies (IT'09)*. Lithuania, April 23-24, 2009.
- [7] Nakache, J.P., Confais, J. *Approche pragmatique de la classification : arbres hiérarchiques, partitionnement*. Editions TECHNIP, 2005.

# Software Project Effort Estimation Non Lineal Mathematical Models

Pablo R.Soria<sup>1</sup>, Borja Martín<sup>1</sup>, Marian Fernández de Sevilla<sup>1</sup>,  
María J.Domínguez-Alda<sup>1</sup>, Miguel A.Herranz<sup>1</sup>

<sup>1</sup>CuBIT, Software Measurement Laboratory; Department of Computer Science,  
University of Alcalá, Madrid (Spain)

[pablo.rsoria@uah.es](mailto:pablo.rsoria@uah.es)

**Abstract.** This paper presents a review of the software project estimation methods that have been developed across the software engineering history, mainly focused on the effort estimation non lineal mathematical models. These methods and models have been classified from new criteria and are based specifically on public dissemination models. For each model is showed its main characteristics, elements and equations that allow us to see as a whole the operation and implementation of each of these effort estimation methods.

**Keywords:** Software Engineering, Effort estimation, Project size, Software project planning process.

## 1. Introduction

Since the first computing developments until the current ones, a fundamental problem has been the fulfillment of certain deadlines for delivery within an established cost, as well as to be able to track and control the projects evolution. Therefore, the establishment of some methods that would enable us to obtain these objectives in a way as realistic and accurate as possible has become an increasingly factor for The Computing Science as a whole, and such methods have been based on knowledge acquired by different disciplines of this science, since Software Engineering to Artificial Engineering.

In addition to produce ever better results in the original objectives, the continuing evolution that the estimation methods have experienced, has also allowed to obtain other benefits such as the upgrading of the projects risk analysis or the possibility of quantitative analysis on the effectiveness of different change proposals of the software construction processes.

Since the 1960's until now, have been published a large number of models and have also been proposed various classifications of the same based on different criteria. One of the best-known and referenced in the literature is the one proposal by Conte, Dunsmore and Shen [6] comprising four types of models:

1. *Historic / Experimental*

It refers to those models based on a set of equations proposals by an expert.

2. *Statistics*

It brings together the models using a regression analysis to establish the relationship between the

effort and drivers of cost. It is differed two rates depending on the equations used:

- a. Lineal. The algorithms are lineal equations.
- b. Non Lineal. The algorithms are non lineal equations.

3. *Theoretic*

They are based on theories on how human beings are communicated among themselves, on how the human mind works during the programming process or on mathematical laws that follows the process of developing a software product.

4. *Compounds*

They incorporate a combination of analytical equations, statistical adjustment data and experts ideas.

The rest of this paper is structured as follows; Section 2 provides a new proposed classification for effort estimation model following the basis lines of previous models. Then Section 3 describes the non lineal and public mathematical models that we are going to present and analyze in a chronological order. Finally, conclusions are provided in Section 4.

## 2. New Proposed Classification for Effort Estimation Models

Here we are going to propose a new classification for effort estimation models [7] that although takes some ideas

of the proposed previously, presents a new approach under which such methods can be structured into four groups:

1. *Parametric Mathematical Models*

This group would include all those models using mathematical equations to make these estimates. The difference in whether they have been generated by experts is not included because it is assumed that the intervention of experts is essential in the development of the equation. Nor is difference between tabular models, multiplicative, lineal and non lineal. The reasons are from one hand that multiplicative and lineal equations are considered a particular case of the non lineal ones and on the other hand, tabular models or that ones that used lineal equations are already in disuse and all the new published methods use non lineal equations.

Only the differentiation between Public Models and Owners Models is introduced due to its important consideration, because on the first ones will be possible to make a research work and study, as well as an adequacy to a local environment on the part of the researches or users; and on the second ones will be only possible to know in a general way its functioning or employment method and procedure to acquire or rented. Due to this reason, we will focus our study on the public methods.

2. *Estimation based on Experts Experience*

Models classified within this method are those that provide a way to extract of the most objective as possible knowledge of experts to carry out the estimations. This is normally done through a set of steps or through the use of questionnaires.

3. *Learning Aimed Techniques*

This method is based on the use of data collected in previous projects to make these estimates and in this sense is similar to the Mathematical Models, which also used data from previous projects to determine its parameters. But the difference that present models classified in this methodology, a part of not being based on mathematical equations, is that does not pretend to be of universal application but his goal is to make precise estimates for those projects of the same type of those who have been used to learn.

4. *Dynamic Models*

These models assume that the cost factors of a software project change over time and estimates and make estimates and simulations of such variations. Therefore the basis of this method is very different from the other three, which have a static view of the system.

This paper will focus on the non lineal parametric mathematical estimation models particularly on the public disseminated ones, due to its wide use and transparency.

### 3. Mathematical Models

Mathematical models are based on the development of some mathematical equations that allow, in most universal way as possible, to obtain, from the introduction of a set of an independent variables values such as the size of software product to develop or the users experience; the value of another set of dependent variables, such as the effort or the time needed to develop the product. These equations, which began being lineal in the first models that were developed and that in the latest models are non-lineal, are adjusted through a set of parameters, which, in turn are those that determine the scope of the model.

The development of the parametric mathematical models began in the mid 1960's and since then, a large amount of models have been developed until now. This model's evolution has been linked to the progress that has occurred in the age of the computers, with the constant change that has taken place in the hardware and software and especially in the way to perform this last one.

Mathematical models can be classified in public and owners or patented. The first models that were developed used to be public since their equations, parameters, etc., were free dissemination published by the authors, which has allowed its study, analysis and subsequent publications of it, as well as its utilization and calibrated in different environments in which were designed. But as this field took importance, was generating commercial interests, so that among the latest models used in recent years, most of them are owners or patented by companies that operate them. Although these patented models acted as a black box and we can only know their results, most of them tend to be the result of the evolution of some previously free model so it is assumed that their fundamental bases are the mathematical equations in which was based on the original model.

In this paper we are going to focus our research only on public models due to its extensively and wide range detail, all this in a chronological order.

#### 3.1 GRC

The official name for this model is *GRC* that comes from General Research Corporation, the company that developed it. It was first time published in 1974 by Taback [11].

This model calculates the cost of the system in dollars straight from the number of instructions, without making a previous effort calculation using the following equation for Effort:

$$c = 0.232(s)^{1.43} \quad (1)$$

Where  $c$  represents the cost measured in dollars,  $s$  the size of the system in SLOC<sup>1</sup>.

---

<sup>1</sup> SLOC – Source Lines of Code

### 3.2 Doty

The official name for this model is *Doty* that comes from Doty Associates, the contractor company in which was carried out the study entrusted by Rome Air Development Center (RADC) of the Data and Analysis Center for Software (DACS) of the Department of Defense (DoD) of the United States of America, who allowed this model development. It was first time published in 1977 by Herd et al. [8].

This model estimates the effort, costs and development for a certain software project. In order to choose the constant values of the calculation equation, the Doty model distinguishes four types of programs or program modules:

1. On-line or Interactive
2. Real Time or Control
3. Calculation
4. Operating Systems

The size of the proposed system is estimated by comparison with similar systems. One of the conclusions introduced by this model is that the time needed to write a given number of instructions in a high-level language is the same that to write them in an assembly language, but as the high-level language is smaller, normally in a relation from 3 to 7, the result is that the productivity increases. In addition to the fact that the high-level language is higher in clarity and therefore it facilitates the maintenance.

This model is based on the following equations for Effort:

$$\text{On-line:} \quad e = 4.495(s)^{1.068} \quad (2)$$

$$\text{Real Time:} \quad e = 4.573(s)^{1.228} \quad (3)$$

$$\text{Calculation:} \quad e = 2.895(s)^{0.784} \quad (4)$$

$$\text{Operating Systems:} \quad e = 12.039(s)^{0.719} \quad (5)$$

Where  $e$  represents the Effort measured in MM<sup>2</sup> and  $s$  the size in SLOC.

### 3.3 Aerospace

The official name for this model is *Aerospace Model*. It was first time published in 1977 by James [9].

This model estimates separately the effort for programs in real time and the rest using the following equations for effort estimation:

$$\text{Real Time:} \quad e = 0.057 (s)^{0.94} \quad (6)$$

$$\text{Other programs:} \quad e = 2.012 (s)^{0.404} \quad (7)$$

Where  $e$  represents the Effort measured in MM and  $s$  size in SLOC.

### 3.4 IBM – FSD - Walston-Felix

The official name for this model is *IBM-FSD* that comes from International Business Machines Federal Systems Division, the company in which was carried out the study that allowed the development of this model. It is also known as *Model of Waltson-Felix* because it was first published in 1977 by Walston and Felix [13].

Although this model determines the effort on the basis of the lines of code produced; the analysis of the database used by the authors propose another series of relations: productivity versus percentage of new code developed, documentation versus code developed, time versus code developed, time versus effort, development team versus effort, cost versus code developed, or cost versus effort.

This model is based using the following equation for Effort and Development Time:

$$\text{Effort:} \quad e = 5.2(s)^{0.91} \quad (8)$$

Where  $e$  represents the effort measured in MM,  $s$  the final size in thousands of SLOC.

$$\text{Development Time:} \quad t = 2.47(e)^{0.35} \quad (9)$$

Where  $t$  represents the time developed in months and  $e$  the effort measured in MM.

### 3.5 Bailey-Basili

The official name for this model is *Baylei-Basili Model*. It was first time published in 1981 by Bailey and Basili [2].

More than a model to estimate, Bailey and Basili presented a construction method of a local model to estimate. The process of model generation consists of three steps:

1. Calculate the effort from the original equation provided by the model.
2. Determine what combination of factors differs to the project which has been estimated and that could explain variations between the data measured in the actual project and the estimated values obtained through the original equation. Bailey and Basili identified about 100 attributes dependent on the local development environment as possible causes of the variation between the effort measured and the effort estimated, although working only with 18 sets of data could not consider mathematically so many attributes. In order to solve this problem, they proposed the use of different techniques such as the experience of experts, or the correlation matrix, in order to select the most influential attributes; in this way 21 attributes were finally obtained. In addition

<sup>2</sup> MM – Men per Month

they grouped these attributes following the idea that the group had a positive or negative impact on the effort and that was easily understandable. The three groups obtained were:

1. Total Methodology (METH)
  1. Tree Diagrams.
  2. Top – Down Design.
  3. Design Formalism.
  4. Code Reading.
  5. Head of Programmers Group.
  6. Formal Test Plans.
  7. Development Units.
  8. Formal Training Plans.
2. Accumulated Complexity (CMPLX)
  1. User Interface Complexity.
  2. Changes from the client once initiated the design.
  3. Implementation Process Complexity.
  4. Complexity of the Program Flow of Data.
  5. Internal Communication Complexity.
  6. External Communication Complexity.
  7. Database Complexity.
3. Accumulated Experience (CEXP)
  1. Aptitude of the programmer.
  2. Programmer Experience with the machine.
  3. Programmer Experience with the language.
  4. Programmer Experience with the application.
  5. Previous work of the Team working together.

Each of these groups is evaluated in a range of 1 to 5.

3. Use the model to predict new projects.

The original equation or basic relationship between effort and size was determined, as has been mentioned above, using 18 sets of data from the SEL (Software Engineering Laboratory) of the NASA (National Agency for the Administration of Space). The following equation was used to estimate the effort:

$$e = 3.5 + 0.73s^{1.16} \prod_{i=1}^3 x_i \quad (10)$$

Where  $e$  represents the Effort measured in MM,  $s$  size in SLOC, and  $x_i$  the value of each group of attributes.

### 3.6 COCOMO

The official name for this model is *COCOMO* that comes from CONstructive COSt MOdel, and it has two versions. The first one is called COCOMO 81; the name 81 takes of the last two digits of the publication year and added when the new version COCOMO II is launched. COCOMO 81 was first time published in 1981 by Boehm.

It was developed by Bary Boehm [3] in TRW, based on the analysis of 63 completed software projects and had three submodels:

1. *Basic*.
2. *Intermediate*.
3. *Detailed*.

COCOMO 81 model is based on the following equations of Effort and Development Time:

$$\text{Effort:} \quad e = \sum_{i=1}^n [a(s)^b \prod_{ij=1}^{15} x_{ij}] \quad (11)$$

Where  $e$  represents the Effort measured in MM,  $s$  size in SLOC,  $a$  and  $b$  are constant, and  $x_{ij}$  the value of driver of cost  $j$  for phase  $i$ , where  $i$  can assert 1 for the model intermediate and between 1 and 4 for the detailed model.

$$\text{Development Time:} \quad t = c(e)^d \quad (12)$$

Where  $t$  represents the time in months,  $e$  represents the effort measured in MM, and  $c$  and  $d$  are constant.

The second version, COCOMO II, was published in 1995 by Boehm [4] and consists of the following Tools:

1. COCOMO II. 1999.0. Developed by the team of the University of Southern California that has developed the theoretical model. It is free of charge.

[ftp://ftp.usc.edu/pub/soft\\_engineering/COCOMOII/cocomo99.0/c990windows.exe](ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo99.0/c990windows.exe)

2. COSTAR. Developed by Costar. It is an interactive tool that allows monitoring of projects, as well as experiments of the type: What would happen if?

Web: <http://www.softstarsystems.com>

Web: <http://sunset.usc.edu/COCOMOII>

This model is composed by three submodels:

1. *Composition of Applications*

This submodel is used to estimate the effort and the time of development in projects using CASE (Computer Aided Software Engineering) tools for rapid application development. Although these products are very different, are enough simple to be rapidly built from interoperable components. Typical components are the builders of GUI (Graphical User Interface), database or objects managers, transaction processing, etc. as well as components of specific domains such as medical or industrial control packages.

This model uses as variable to measure the size of the product The Object Points [1], [10]. The object points are basically a count of the windows, reports and modules, developed in a language of third generation, in the application. Each count is weighted by a complicating factor of three levels; simple, medium and complex. There is only one calibration of this model [10] due to the lack of data.

## 2. Initial design

This sub-model takes into account the exploration of different architectures of the system and operation concepts. Normally this is not enough for making precision estimations.

It utilizes as product size the Function Points or Source Lines of Code, when available.

It introduces a set of 5 Scale Factors. Each one received a value that will be a real number corresponding to the range in which the factor of project that is being developing is placed. There are six ranges: Very Low, Low, Nominal, High, Very High and Extra High. Scale Factors are the following:

1. Precedents (PREC)  
It takes into account the experience of the organization in the development of this type of applications.
2. Flexibility in the development (FLEX)  
It takes into account the rigidity of the requirements and the constraints. Along with the previous one, they substitute the COCOMO 81 development models.
3. Architecture / Risk Solution (RESL)  
It takes into account the measures taken for the deletion of risks.
4. Team Cohesion (TEAM)  
It reflects the cohesion and synchronization difficulties of the ones implicated in the project, due to its diverse precedence and objectives. These are: users, customers, developers, maintenance team, etc.
5. Processes Maturity (PMAT)  
This factor takes into account the maturity level of the organization processes, for which it utilizes the five levels scale of the CMM (Capability Maturity Model) developed by SEI (Software Engineering Institute) in the Carnegie Mellon University.

Furthermore, this sub model utilizes 7 Effort Multipliers which receive a real number taking the range into account, of six levels (Very Low, Low, Nominal, High, Very High and Extra High), in which the multiplier is located for the studied project. The effort multipliers value table is calibrated joining the calibrated effort multipliers for the Post – Architecture model (USC-CSE 1997). The aforementioned drivers of cost are:

1. Personnel Capability (PERS)
2. Reliability and complexity of the product (RELY)
3. Reuse required (RUSE)
4. Platform difficulty (PDIF)
5. Personnel experience (PREX)
6. Facilities (FCIL)
7. Schedule (SCED)

## 3. Post – Architecture

This sub model can be utilized when the high level design has been completed and when we have got detailed information of the model and, as its name suggest, the software architecture is well defined and established. It allows making estimations for the life cycle set of development and it is an extension of the COCOMO 81 intermediate model. It utilizes Function Points and / or Source Lines of Code as an input for the product size parameter. It utilizes a set of five scale factors, equals in its conceptual foundation to the previously ones seen in the initial design sub model, and seventeen effort multipliers, each one with a range of six equal levels same as the ones seen for the initial design sub model. This sub model has been calibrated on a 161 projects database compiled of the commercial industry, aerospace, governmental and non-profit organizations; using a Bayesian approximation [5]. The drivers of cost are grouped in the following four categories:

1. Product
  1. Software required reliability (RELY)
  2. Database size (DATE)
  3. Product complexity (CPLX)
  4. Reuse required (RUSE)
  5. Developed documentation (DOCU)
2. Platform
  6. Run time constraints (TIME)
  7. Main store constraints (STOR)
  8. Platform volatility (PVOL)
3. Staff
  9. Analysts aptitude (ACAP)
  10. Programmers aptitude (PCAP)
  11. Experience in the similar applications development (AEXP)
  12. Experience in the platform development (PEXP)
  13. Experience with the language and tool (LEXP)
  14. Personnel continuity (PCON)
4. Project
  15. Software tools Utilization (TOOL)
  16. Multiple site developments (SITE)
  17. Necessary time for development (SCED)

COCOMO II model is based on the following equations for Effort and Development Time:

$$\text{Effort: } e = a.(s)^{(b+c \sum_{i=1}^5 y_i)} . \left( \prod_{j=1}^n x_j \right) \quad (13)$$

Where  $e$  represents the effort measured in MM,  $s$  the size in SLOC,  $a$ ,  $b$  and  $c$  are constants, and  $x_j$  is the value of the driver of cost  $j$ ,  $n$  is equal to 7 in the preliminary design sub model and 17 in post – architecture, and  $y_i$  is the scale factor  $i$ .



$$\begin{array}{l} \text{Develop.} \\ \text{Time:} \end{array} \quad t = (e)^{0.33+0.2(b-1.01)} \cdot \left(\frac{SCED \%}{100}\right) \quad (14)$$

Where  $t$  represents the time in months,  $e$  represents the effort measured in MM,  $b$  is a constant and  $SCED$  is the effort multiplier Time needed for development.

### 3.7 COPMO

The official name for this model is *COPMO* that comes from COoperative Programming MOdel. It was first time published in 1984 by Thebaut and Shen [12].

This model uses as independent variables the size of the product to be developed and the measurement of the team size that develops the product. It is based on the following equation for Effort:

$$e = a + b.s + c(n)^d \quad (15)$$

Where  $e$  represents the Effort measured in MM,  $s$  the size in thousands of SLOC,  $a$ ,  $b$ ,  $c$  and  $d$  are constants, and  $n$  is the average size of the development team.

## 4. Conclusions

In this paper, we have presented a review of the software project effort estimation methods that have been developed across the software engineering history, mainly focused on the effort estimation non lineal mathematical models. With this review, we have tried to show how these estimation models have evolved from its origins and the mathematical bases that have followed for its implementation.

Taking a clear view of how these estimation models have been based, then may understand how are developing and improving the current ones and lines we can follow to analyze the next.

This is not attempts to bring a new model or assess, it is just a simply intend to present a chronological evolution according to the new classification on software project effort estimation Non Lineal models with public dissemination that we have contributed. Therefore, as futures lines we will assist in upcoming revisions to the owner models which have been developed so far and that have led to a great progress in the improvement of the processes of effort estimation in the software project planning process.

## Acknowledgement

We would like to thank the University of Alcalá for supporting this research (Ph.DC researchers support programme).

## References

- [1] Banker, R., Kauffman, R. and Kumar, R.: An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment, In *Journal of Management Information System*, 1994.
- [2] Baylei, J. and Basili, V.: A Meta-model for Software Development Resource Expenditures, In *Proceedings of the Fifth International Conference on Software Engineering*, (1981), pp. 107-116.
- [3] Boehm, B.: *Software Engineering Economics*, Editorial Prentice Hall, 1981.
- [4] Boehm, B., Clark, B., Horowitz, E., Madachy, R., Selby, R. and Westland, C.: Cost Model for Future Software Life Cycle Processes: COCOMO 2.0, In *Annals of Software Engineering Special Volume on Software Process and Product Measurement*, Eds. J.D. Arthur, S.M. Henry and J.C. Baltzer, Edit. AG Science Publishers, Amsterdam (Holland), (1995), Vol. 1.
- [5] Chulani, S., Boehm, B. and Steece, B.: Calibrating Software Cost Models Using Bayesian Analysis, En *Technical Report USC-CSE-98-508*, (1998).
- [6] Conte, S. D., Dunsmore, H. E., Shen, V. Y.: *Software Engineering Metrics and Models*. Benjamin / Cumming Co., Inc. Menlo Park. 1986.
- [7] Cuadrado-Gallego, J. J.: Métodos de Estimación de Proyectos Software, In *UC3M-TR-CS-2000-01* Vol. 1, Ed. Dep.Informática UC3M, (Spain), 2000.
- [8] Herd, J., Postak, J., Russel, W. and Stewart, K.: Software Cost Estimation Study – Study Results, In *Final Technical Report, RADC-TR-77-220*, Ed. Doty Associates, Inc., 1977.
- [9] James, T. Jr.: Software Cost Estimating Methodology. In *IEE Proceedings of National Aerospace Electronic Conference*, (1997), pp 22-28.
- [10] Kauffman, R. and Kumar, R.: *Modelling Estimation Expertise in Object Based ICASE Environments*, Editorial Stern School of Business Report, New York University, January 1993.
- [11] Taback, R., and Ditimore, J.: Estimation Computer Requirements and Software Development Costs, In *RM-1873*, Eds. General Research Corporation, 1974.
- [12] Thebaut, S. y Shen, V.: An Analytic Resource Model for Large-Scale Software Development, In *Inf. Proc. Management*, 20 (1-2), (1984).
- [13] Walston, C. and Felix, C.: (1) A Method of programming Measurement and Estimation, In *IBM System Journal*, 16, (1), (1977), pp. 54-73. (2) Author's Response, In *IBM System Journal*, 16 (4), (1977), pp. 422-423.

# Software Estimation: Universal Models or Multiple Models?

Alain Abran<sup>1</sup>, Juan Jose Cuadrado Gallego<sup>2</sup>

<sup>1</sup>École de Technologies supérieures – Université du Québec, Montréal, Canada

[alain.abran@etsmtl.ca](mailto:alain.abran@etsmtl.ca)

<sup>2</sup>Universidad de Alcalá de Henarès, Madrid Spain [jjcg@uah.es](mailto:jjcg@uah.es)

**Abstract.** In the field, there is a very large diversity of development processes in use, and various mixes of cost drivers, each with a different impact depending on the context. The classical approach to building estimation models in software engineering is to build a single estimation model and include within it as many cost factors (i.e. independent variables) as possible. In this paper, we do not postulate that there exists a single estimation model that is ideal in all circumstances, but rather we report on exploratory research conducted over the past few years looking at relevant concepts from the field of economics and from discussions with organizations attempting to understand the data that they have collected on their projects. The purpose of exploratory research is not to demonstrate a hypothesis, but to identify new potentially relevant concepts to develop hypotheses to be tested later on with empirical or experimental data.

## Categories and Subject Descriptors

D2.9 [Management]: Productivity

## General Terms

Measurement

## Keywords

Process measurement, Productivity measurement, Estimation

## 1. INTRODUCTION

The classical approach to building estimation models in software engineering is to build a single estimation model and include within it as many cost factors (i.e. independent variables) as possible.

**A- Model based on completed projects:** When the builders of estimation models have access to a reasonable set of completed projects, they typically attempt to build a single model for all of these projects which takes into account the largest possible number of the variables included in their data repository. This approach is best illustrated with the design of the COCOMO models [1-3], containing a large number of cost drivers, with:

- the authors' own definition of these cost drivers,
- the authors' own measurement rules for these cost drivers and their own assignment of impact factors for each of them.

This, of course, leads to complex models with a large number of variables, but seldom with enough data points for meaningful statistical analysis or the confidence that

such models can be used in environments other than the one for which they were developed initially.

**B- Models based on opinions:** Another approach is to build models based on the authors' opinions about the variables and their estimation of the impact on a model's behavior. With such an approach, it is very easy to come up with any number of new cost drivers: being based on opinion only, there is no cost for data collection and analysis. This can be observed in some of the 'use case points'-based models [4-7]. Furthermore, for many of such models – some available free from the web, there has not even been any attempt to demonstrate how well they perform, even within the context in which they were built.

Models built without data (or with not enough data) and those that include many opinion-based cost drivers (i.e. independent variables) lead the managers to believe that the majority of the important cost drivers have been duly taken into account by the models: the managers are then led to believe that, by using these models, they reduce the risks inherent in estimation. This makes them feel good, but falsely so, since such models are not supported by empirical evidence, and their limitations have not been documented. Moreover, lured by that 'feel good' potential, managers may find themselves dealing with even more uncertainty.

Over the past 30 years of research on software project estimation, expert practitioners and researchers have come up with many models with different mixes of cost drivers, but with little commonality, and to date most of them have not been generalized to contexts other than the one on which they were based.

In this paper, we report on exploratory research conducted over the past few years looking at relevant concepts from the field of economics and from discussions with organizations attempting to understand the data that they have collected on their projects. The purpose of exploratory research is not to demonstrate a hypothesis, but to identify new potentially relevant concepts to develop hypotheses to be tested later on with empirical or experimental data.

This paper is organized as follows. Section 2 presents a few economics concepts used to model a production process, and corresponding characteristics that may be relevant to build multiple models and interpret them. Section 3

presents next an approach to build distinct models by size ranges.

In this paper, we do not postulate that there exists a single estimation model that can be considered ideal in all circumstances. Rather, we look for concepts and approaches which could contribute to the identification of distinct models corresponding to distinct production processes.

## 2. PRODUCTION MODELS: SOME CHARACTERISTICS

### 2.1 A production process

How can the performance of a development process be estimated in the future if its current and past performance, and the variations in that performance, are not known?

- What is a development process?
- How is its performance determined?
- How can a development process be modeled to build estimation models?

A development process can be modeled as a production process: this can be illustrated in its most simplified form with three main components – see Figure 1: Inputs, Activities within the process itself (in software, this corresponds to the development life cycle selected and implemented), and Outputs (the software itself and the environment in which it will be executed).

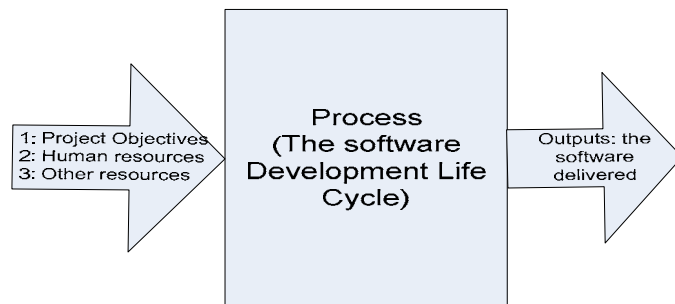


Figure 1: A production process

The inputs to a production can typically be classified into three groups:

- 1- The objectives for a specific production run: In software engineering, this corresponds to the objectives of the software development project. These objectives are often stated in terms of functional requirements and non functional requirements for the software to be delivered, as well as in terms of the project priorities (costs, quality, duration).
- 2- The human resources to be made available for the development process, that is, the staff who will be available to work on the development project and carry out all the tasks of the sub processes within the

process. These inputs are typically measured in work-hours (or person-days/-weeks/-months).

- 3- Any other non human resources available for carrying out the tasks, such as the technical environment, including the hardware-software platform, the tools, the methodologies, etc.

### 2.2 Productions models with fixed and variable costs

A production model is typically built with data from projects completed, that is, when:

- all the information on a project is available;
- there is no more uncertainty on either the inputs or the outputs: all the software functions have been delivered; and
- all the hours worked on the project have been accurately entered into a time reporting system.

The points on the graph in Figure 2 represent, then, the number of hours it took to deliver the corresponding functional size of the projects completed.

- The x axis represents the functional size of the software projects completed;
- The y axis represents the effort in number of hours that it took to deliver a software project.

The straight line across Figure 2 represents a statistical model of the production process and is based on the performance of past projects.

This linear model models the relationship between effort and size, and is represented by the following formula:

$Y$  (effort in hours) =  $f(x) = a \times \text{Size} + b$  where:

- Size = number of Function Points (FP)
- $a$  = variable cost = number of hours per Function Point (hours/FP)
- $b$  = fixed cost in hours

In terms of units, this equation then gives:

$$Y \text{ (hours)} = (\text{hours/FP}) \times \text{FP} + \text{hours} = \text{hours}$$

In a production process, there are typically two major types of costs incurred to produce different sets of the same types of outputs:

Fixed costs: the portion of the resources expended (i.e. inputs) that does not depend on the number of outputs. In Figure 2, this corresponds to  $b$ , the constant in hours at the origin when size = 0.

- Example of a fixed cost: at  $x = 0$ ,  $b$  represents the fixed cost of this production process (i.e. in this organization, a cost of  $b$  hours of project effort is required to set up and manage a project independently of its size).

Variable costs: the portion of the resources expended (i.e. inputs) that depends directly on the number of outputs produced. In Figure 2, this corresponds to the slope of the model, that is: slope =  $a$  in terms of hours per function point

(that is, the number of work hours required to produce an additional unit of output, that is, the independent variable  $x$ ).

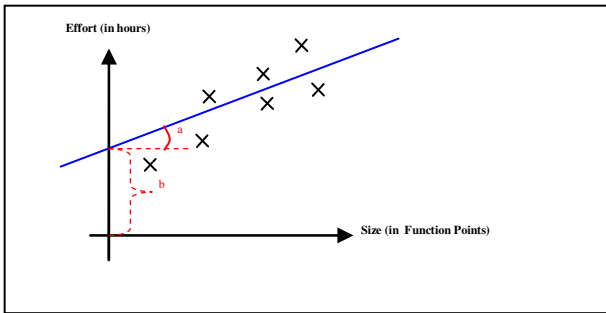


Figure 2: Production model: fixed & variable costs

### 2.3 Wedge-shaped datasets

A graphical representation of project datasets in software engineering typically has the wedge-shape distribution illustrated in Figure 3 [8-9]. It can be observed in this figure that, as the project effort increases on the x axis, there is a corresponding larger dispersion of the data points across the vertical axis: for projects of similar effort, there are increasingly wide variations of project duration on the y axis as the project effort increases. This is often referred to as a wedge-shaped dataset. This had initially been observed by [10-11], and it is typical of most data subsets built with data from large repositories (such as illustrated in [8]).

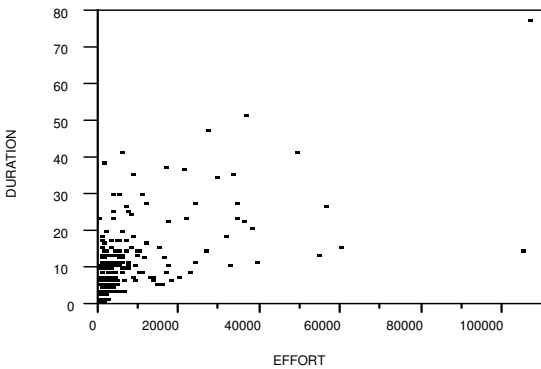


Figure 3: Example of a wedge-shaped dataset [8].

### 2.4 Low & high sensitivity to functional size: multiple models?

In production processes, there might be ones where:

- 1 additional unit of output requires exactly 1 additional unit of input,
- 1 additional unit of output requires less than one additional unit of input, and
- 1 additional unit of output requires more than one additional unit of input.

When the increase in output units requires a correspondingly smaller increase in the number of input units, the production process is said to have lower

sensitivity to size: the larger the number of units produced, the more productive the production process.

By contrast, when an increase in output units requires a larger increase in the number of units for each additional output, then the production process is said to have high sensitivity to size: for each additional unit produced, the less productive the production.

Let us revisit the typical pattern of wedge-shaped datasets of software projects – see Figures 3 and 4. When looked at with the analytical grid of the concepts of low and high sensitivity to size, this single wedge-shaped dataset can be decomposed into three subsets, as follows – see Figure 4:

- Zone 1: The lower part of the wedge-shaped dataset. This lower part represents the set of projects demonstrating little sensitivity to increases in size: indeed, for this subset, even large increases in size do not lead to noticeably correspondingly large increases in effort. In practice, it is as if, in this subset, the effort required is almost insensitive to an increase in the number of functions in the software being developed.
- Zone 3: The upper part of the wedge-shaped dataset. This upper part represents the set of projects demonstrating high sensitivity with respect to functional size as the independent variable (that is, a small increase in size requires a much larger increase in effort – in either fixed or variable costs, or both).
- Zone 2: Finally, there is sometimes a third dataset that is somewhere in the middle range of the wedge-shaped dataset

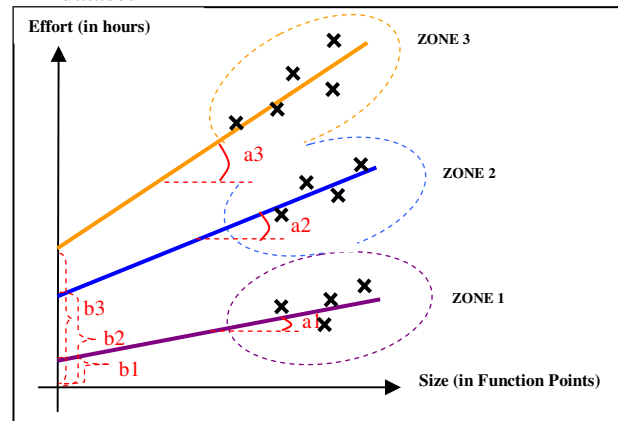


Figure 4: Wedge-shaped dataset with distinct sensitivities to functional size

This leads, then, to three distinct production models (often referred to as ‘estimation models’ in the software engineering literature):

$$f_1(x) = a_1 * x + b_1, \text{ which corresponds to a data sample in zone 1.}$$

$f_2(x) = a_2 * x + b_2$ , which corresponds to a data sample in zone 2.

$f_3(x) = a_3 * x + b_3$ , which corresponds to a data sample in zone 3.

Each of these 3 models has its own slope (the  $a_i$ ), as well as its own fixed costs  $b_i$ . The next question is, of course, what causes these different behaviors?

Of course, the answer cannot be found by graphical analysis alone, as there is only a single independent variable in a two-dimensional graph. This single variable does not provide, by itself, any information about the other variables, or about similar or distinct characteristics of the completed projects for which data are available.

However, the projects included within each subset can be identified nominally by the organizations having collected such data [9]. Each project within each subset should next be analyzed to figure out:

- which of their characteristics (or cost drivers) have similar values within the same subset; and
- which characteristics have very dissimilar values across the 2 (or 3) subsets.

Of course, some of these values can be categories (on a 'nominal' scale type: for example, a specific Data Base Management System (DBMS) has been used for a subset of projects, etc.).

The ability to discover the different values of such characteristics can then be used to characterize such datasets, and to set the parameters for selecting which of these three production models to use later on for estimation purposes.

### 3. DISTINCT MODELS BY SIZE RANGE

We have often observed in organizations measuring the size of their projects the following:

- a large number of small projects, a smaller number of medium size projects and, in comparison, few much larger projects [12].
- a lightweight development process for small projects, a very heavyweight development process for very large projects and a tailored process for the medium-size projects.

Notwithstanding this, the usual approach in software engineering is to look for a single model across all the size ranges. An alternative approach is to build estimation models by segregating the available dataset into ranges of project sizes when an organization has different processes by size ranges.

When such information about the organizational processes is not known, a similar analysis can be done by segregating

by the data set by density distributions in contrast to the general practice of building a single model for the whole range of data points [13]. Indeed, a single model across the full range is not a strict requirement of the statistical techniques used to build such models; if this is the case, users should beware, since the levels of confidence may vary across the ranges when the data is not normally distributed)..

These concepts are illustrated in Figure 5 with an exemplary data set with a large number of small projects (e.g. fewer than 100 CPF), while there are only a few projects in the much wider interval for large projects. Of course, a single estimation model can be built with this full dataset. However, another approach would be to build multiple regression models representative of the density of the data points that can be identified graphically [13].

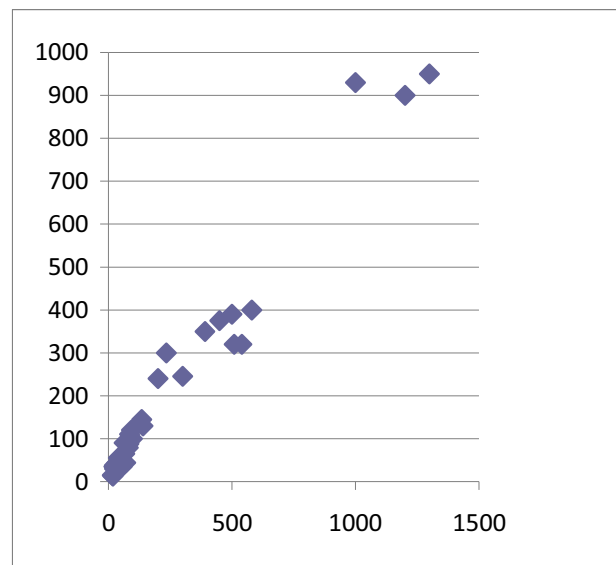


Figure 5: Dispersion of data points

For example, in Figure 5, there are:

- about 20 small projects within the 15 to 150 FP range
- about 10 projects within the 200+ to 600 range
- only 3 projects within the 1,000 to 1,300 FP range

For this specific dataset, it would be much better to build two estimation models, one for the very small projects (Figure 6) and one for the mid-range projects (Figure 7), and to consider the 3 largest projects as an analogy base without statistical strength. This would be preferable to building a single estimation model over such a large range of projects with distinct project densities, and more representative of the projects themselves.

For the small projects of this illustrative set of Figure 5, the regression equation is (Figure 6) :

Effort = 1.01\*FP +3 with an  $R^2$  of 0.87.

For these small projects, the fixed cost is low, that is = 3, and the variable cost is close to 1.

For the mid-sized projects (within this illustrative dataset, of course). the equation is (Figure 7):

Effort = 0.32\*FP + 192 with an R<sup>2</sup> of 0.59,

but with a much higher fixed cost of 192 and a lower slope of 0.32, instead of the steeper variable cost of 1.01 for the small projects. Also, with a slope of 0.32, it exhibits much lower sensitivity to an increase in size within that size range than much smaller projects.

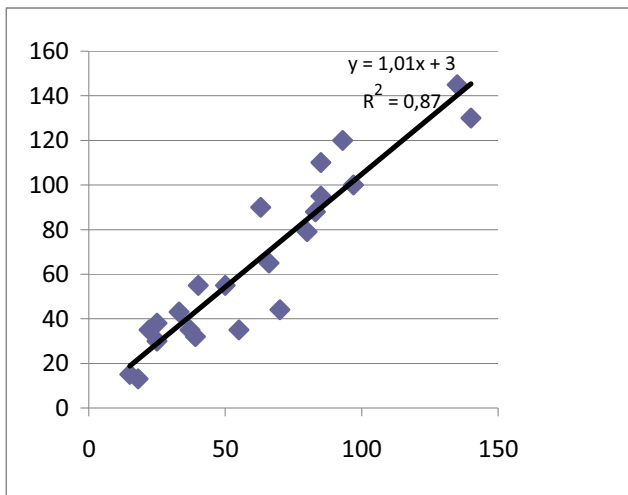


Figure 6: Regression model for the 15 to 150 FP interval

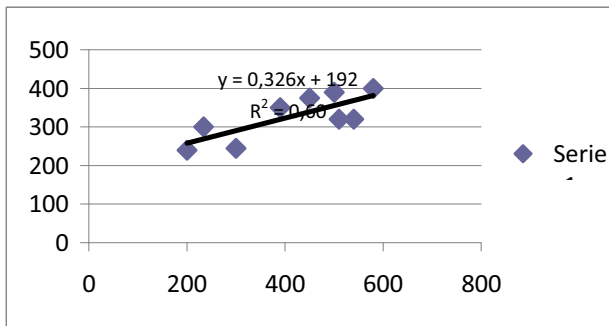


Figure 7: Regression model for the 200 to 600 FP interval

For example, for the estimation of a small project of 50 FP, equation A is highly preferable to the general equation:

- Equation A is built with small projects only, and is therefore much more representative of the project being estimated. Moreover, this equation is not influenced by much larger projects.

For the estimation of a mid-range project of 500 FP, equation B is preferable to the general equation:

- Equation B is built with mid-range projects only, and is therefore much more representative of the one being estimated. This equation is not influenced either by the very small projects or by the much larger projects. Caution must still be exercised, however, because of the limited number of projects within that range to build the estimation model for this range.

For the estimation of a very large project of 1000 FP, there is no generalization significance to Equation B, although these 3 data points can still be used for analogy purposes.

For purposes of comparison, the single model with the single equation for the full dataset is presented in Figure 8. For the full set of projects, the equation is Effort = 0.748\*FP + 22 with an R<sup>2</sup> of 0.967.

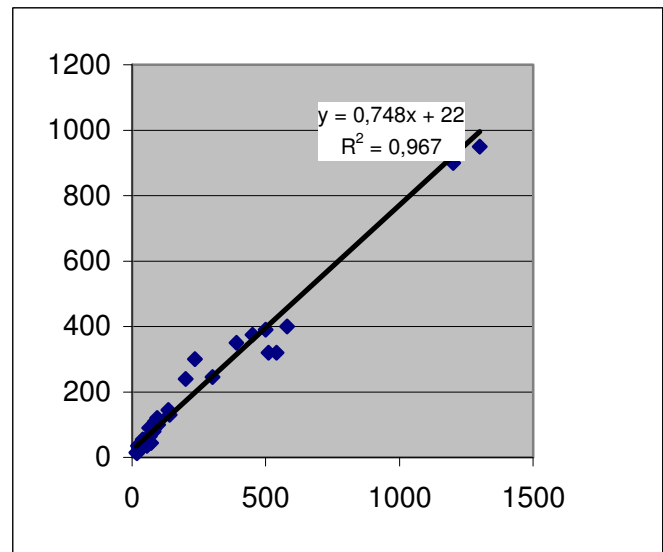


Figure 8: Model with the full dataset

While this model appears to have a better R<sup>2</sup> of 0.967, it is influenced too much by the three largest projects and is not representative of the majority of much smaller projects; therefore, calculation of the magnitude of the relative error (MRE) would lead to a larger MRE for the model of the full dataset, and smaller ones for the models per subset of projects within the size intervals identified.

Also, for the full dataset, the normality distribution of the data is not met, and its regression is correspondingly less statistically meaningful. By contrast, within the two size intervals identified, they would be closer to a normal distribution – within their ranges, of course.

#### 4. SUMMARY

In the field, there is a very large diversity of development processes, and different mixes of costs drivers, each with a different impact depending on context.

Over the past 30 years of research on software project estimation, expert practitioners and researchers have come up with different models with different mixes of cost drivers, but with little commonality, and to date most of them have not been generalized to contexts other than the one on which they were based.

In this paper, we have reported on exploratory research looking at relevant concepts from the economics field and from discussions with organizations attempting to understand the data they have collected on their projects.

The purpose of exploratory research is not to demonstrate a hypothesis but to identify new potentially relevant concepts to develop hypotheses to be tested later on with empirical or experimental data.

In this paper, we did not postulate that there exists a single estimation model that can be considered ideal in all circumstances. Rather, we looked for concepts which could contribute to the identification of distinct models corresponding to distinct production processes.

For instance, section 2 presented a few economics concepts used to model a production process, and corresponding characteristics that may be relevant to software, such as fixed and variable costs as well as production processes with either low or high effort sensitivity to functional size. Section 3 showed another approach to the identification of distinct production models which may manifest themselves across size ranges as organizations adjust project processes as project size increases.

The authors are currently working in collaboration with industrial organizations with datasets similar to the ones discussed in this paper (wedge-shape and with different density of size ranges). Research is in progress to test the contributions of taking into account the various concepts presented in this paper for developing distinct models for the various processes identified by organizations.

## Reference

[1] Boehm, B. W., *Software Engineering Economics*, Englewood Cliffs, NJ, Prentice Hall, 1981.

[2] Boehm, B. W., Abts, C., Brown, A. W.,; Chulani, S., Clark B. K., Horowitz, E., Madachy, R., Reifer, D., and Steece, B., '*Software Cost Estimation with COCOMO II*', Prentice Hall, 2000.

[3] Jørgensen, M. and M. Shepperd., "A Systematic Review of Software Development Cost Estimation Studies", *IEEE Transactions on Software Engineering* 2007, Vol. 33, no. 1, pp.: 33-53.

[4] Clemmons, Roy K., "Project Estimation With Use Case Points," *Crosstalk*, February 2006, pp. 18-22

[5] Mohagheghi, Parastoo, "Effort Estimation of Use Cases for Incremental Large-Scale Software Development," *IEEE International Conference on Software Engineering – ICSE '05*, May 15-21, 2005, St.Louis, MI, USA, pp. 303-311.

[6] Nageswaran, Suresh, "Test Effort Estimation Using Use Case Points," *Quality Week 2001*, San Francisco, CA, USA, June 2001.

[7] Ouwerkerk, J., Abran, A., Evaluation of the Design of Use Case Points (UCP)', *MENSURA2006*, Conference Proceedings edited by the Publish Service of the University of Cádiz [www.uca.es/publicaciones](http://www.uca.es/publicaciones) Nov. 4-5, 2006, Cadiz (Spain), pp. 83-97.

[8] Bourque, P., Oligny, S., Abran, A., Fournier, B., 'Developing Project Duration Models in Software Engineering', *Journal of Computer Science and Technology*, Vol. 22, no. 3, May 2007, pp. 348-357.

[9] Abran, I. Silva, L. Primera, "Field Studies Using Functional Size Measurement in Building Estimation Models for Software Maintenance," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 14, 2002, pp. 31-64.

[10] Kitchenham, B. A., Taylor, N. R., "Software Cost Models," *ICL Technical Journal*, vol. 4, no. 1, May 1984, pp. 73-102.

[11] Kitchenham, B.A., Empirical studies of assumptions that underlie software cost-estimation models. *Information and Software Technology* 34(4):211-218 1992.

[12] Lokan, C., 'What Should you Optimize when building an Estimation Model?', *IEEE International Software Metrics Symposium – METRICS 2005*, 2005.

[13] Abran, A., Ndiaye, I., Bourque, P., "Evaluation of a Black-Box Estimation Tool: A Case Study," In the special issue: "Advances in Measurements for Software Processes Assessment," of the journal *Software Process Improvement and Practice*, vol. 12, no. 2, March-April 2007, pp. 199-218.

# An Empirical Study of the Feedback of the In-process Measurement in a Japanese Consortium-type Software Project

Yoshiki Mitani<sup>1,2</sup>, Tomoko Matsumura<sup>2</sup>, Katsuro Inoue<sup>3</sup>, Mike Barker<sup>2</sup>,  
Akito Monden<sup>2</sup>, Ken-ichi Matsumoto<sup>2</sup>

<sup>1</sup>Information Technology Promotion Agency, Japan/SEC (IPA/SEC), Tokyo, JAPAN, y-mitani@ipa.go.jp

<sup>2</sup>Nara Institute of Science and Technology (NAIST), Nara, JAPAN

{ymitani|tomoko-m|mbarker|akito-m|matumoto}@is.naist.jp

<sup>3</sup>Osaka Univ., Osaka, JAPAN, inoue@ist.osaka-u.ac.jp

## ABSTRACT

Recent research has clarified the usefulness of in-process measurement of a software development project. Past research showed the need for technology to perform in-process measurement of a software development project and utilize the results. Research continues on methods of measurement, analysis, visualization, and feedback of the results, suited to different software development structures. In this paper, the authors present an empirical study focused on requirements for feedback of measurement results. Japanese industry generally uses a hierarchical industrial structure to develop large information systems. In such an environment, effective feedback of measurement results requires different approaches from those that apply to homogenous and flat development organizations. This research illustrates conditions for effective feedback in a hierarchically structured development organization based on actual measurements.

## Categories and Subject Descriptors

D.2.8 [Metrics]: Process Metrics, Product Metrics, D2.9 [Management]: Productivity

## General Terms

Measurement

## Keywords

In-process measurement, Project management, Empirical study

## 1. INTRODUCTION

In-process measurement involves measurement, analysis, and visualization of the state of an ongoing software development project. Research efforts have shown that various technologies are required for in-process measurement. Such research has focused on the structure of the measurement system, the measurement model, and use of case studies to prove the validity of the measurements. [1]-[6] This research applied the Empirical Project Monitor (EPM) and other tools and methods to a typical multi-vendor medium-scale software development project. The authors used interviews with key persons to evaluate the usefulness of in-process measurement in project management and evaluation, and to clarify the requirements for feedback of analysis and visualization results.

This paper briefly summarizes the target project, measurement tools and methods. The authors then discuss the results of project measurement and feedback of project progress based on interviews with key persons after the end of development.

## 2. BACKGROUND OF PROJECT MEASUREMENT

### 2.1 Target Project

The selected target project is a Ministry of Economy, Trade and Industry (METI) funded development of kernel software for an experimental public information system that collects information about traffic conditions and generates useful public information. The project consortium consisted of seven companies, including six major software development companies that are rivals in this field. The project duration is two years, with a first phase of 10 months from basic design to integration test.

### 2.2 Measurement Tools

This project used the Empirical Project Monitor (EPM) and five other measurement tools and methods to collect and analyze process and product data.

#### 1) Empirical Project Monitor measurement and analysis

EPM automatically gathered data from a configuration management system, bug tracking system, and mailing list management system. This process and product data was stored in a relational database, with various analysis and reports available. Figure 1 illustrates some EPM displays.

#### 2) Review report collection

We used an electronic form to collect review reports.

#### 3) Code clone analysis

A code clone is a similar code fragment in the source program. The code clone distribution map displays a bird's-eye view of such fragments in the whole source program.



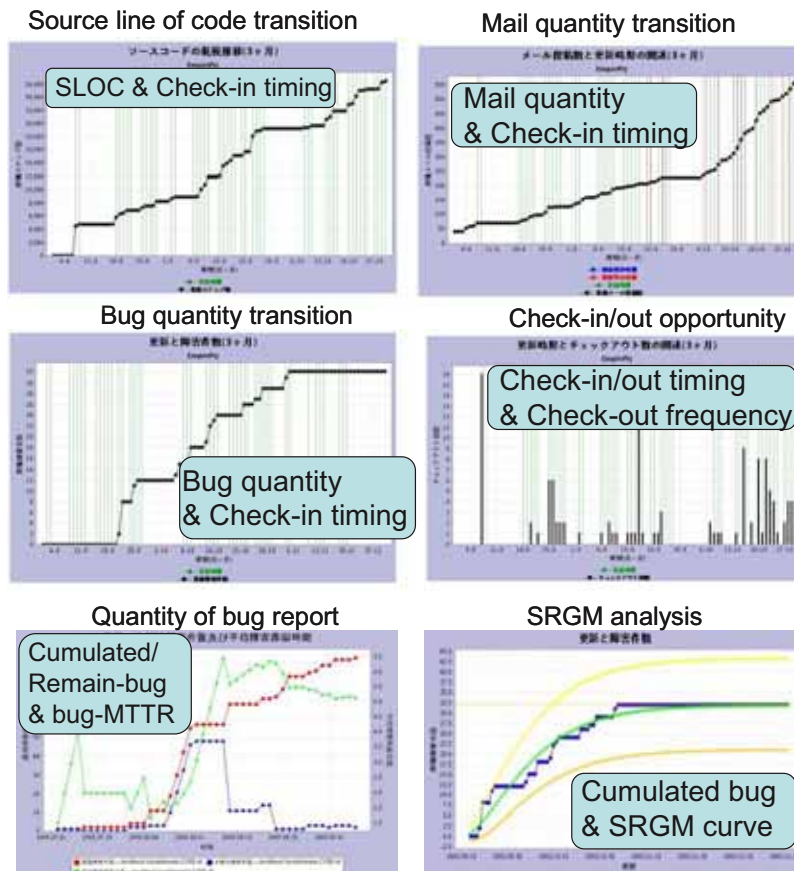


Fig.1. Empirical Project Monitor (EPM) Display Example

4) Benchmark database analysis using a collaborative filtering tool

The project reported benchmark data using a 400 item data sheet developed by IPA/SEC. A collaborative filtering tool identified similar projects from a database of benchmark data from 1000 projects, allowing prediction of future project parameters based on data from the similar projects. [7]

5) Checklist of items for collection of project context data

This research effort defined an interview checklist of 80 items related to project context data.

6) Continuous participation in project meetings for the collection of project context data

To collect project context data, some research staff attended all the project meetings.

### 2.3 Structure of the Project Measurement

#### 2.3.1 Logical View of Measurement Structure

The measurement structure combines a logical structure and the hierarchical structure of the project. Figure 2 shows a logical view of the overall measurement structure. In this logical view, the Software Engineering Center (SEC) in collaboration with the Empirical Approach to Software Engineering (EASE) project collects data from the individual companies.

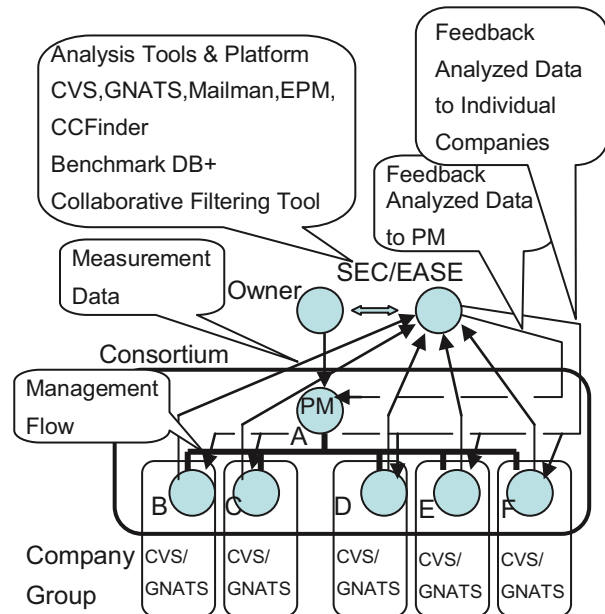


Fig.2. Logical Structure of Project Measurement

The EASE project is a software engineering research project involving both industry and academia. Having this independent group collect the data ensured confidentiality to meet the information privacy requirements of the companies in the consortium.

Each development company uses a CVS server for code management and GNATS server for bug tracking. They periodically share the CVS and GNATS databases with SEC.

During the intercompany integration test phase, a common GNATS server provides bug tracking.

SEC uses various analysis tools such as EPM and provides reports back to the consortium. The SEC/EASE analysts provide the project manager (PM) with overview or summary reports on the project. They also provide individual reports to each development company with details relevant to the company.

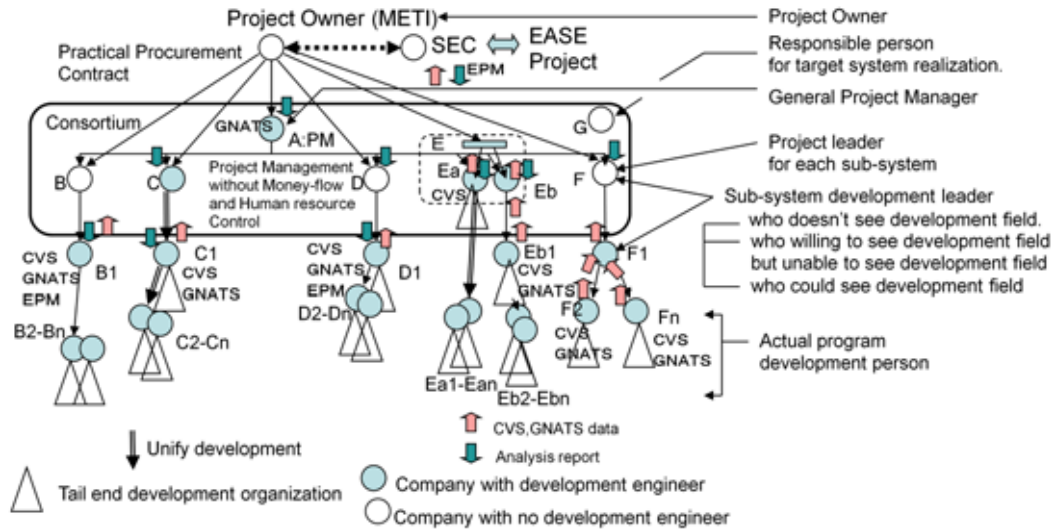


Fig.3 Physical Development Structure

### 2.3.2 Hierarchical View of Measurement Structure

The software industry in Japan has a complicated internal organization involving a strong hierarchical structure. Figure 3 illustrates the overall actual development structure, and Table 1 shows five detailed parts of this structure.

The company groups labeled B to F have various structures for their partner companies. Generally, the structures are confidential except to the project owner, METI. For each group, companies B to F act as main contractors for their own group and manage everything inside their own company group. The main companies contribute requirements definitions and basic designs to the consortium and control intergroup issues. Within a group, there are contracts and financial flows, with developers belonging to specific companies. Quality assurance and product delivery are controlled hierarchically with each group. Subsidiary companies' locations and facilities are separate. The main companies B to F take responsibility for shipping developed software to the owner.

These different structures have the following features:

- 1) There are two kinds of main companies. One has internal software development (ex.B2-Bn,C1). The other does not have internal software development (ex.B,C).
- 2) Generally main companies (ex.B,C) leave practical development work and all responsibilities to their subsidiaries(ex.B1,C1).

- 3) The development process in practice is segmented and performed by individual small software companies using specialized technology (ex.B2-Bn,C2-Cn).
- 4) The company groups used two different approaches to locate the configuration management and bug tracking servers. One approach located the servers at the practical development company (ex.C1,D1). The other approach located the servers at the upper management company that performed delivery to the owner (ex.B1).
- 5) There are various physical relationships between the upper management company and the practical development companies. In one type, all workers on a project are collocated and produce tightly connected products (ex.C,Ea). In another type, while the physical locations are separated, the work is still tightly connected (ex.D,Eb). And in a third type, each company within the group works independently, using periodic business meetings to manage and coordinate the project (ex.B,F).
- 6) In manufacturing or factory type companies, the small software development subsidiaries are not given access to the CVS and GNATS servers of the upper management company, physically located in a large factory (ex.B2-Bn,D1-Dn).
- 7) Some large companies have multiple business divisions, with each division acting as an independent company (ex.E).

**Table 1. Hierarchical Software Industry Structure Example**

Group	B	C	D	E	F
Hierarchical Development Structure					
CVS/GNATS source Feedback target	B1	C1	D1	Ea, Eb1 → Eb	F2 → F1, Fn → F1
Hierarchy Outline	<p>Group company B1 acts as group PM and runs CVS, GNATS and total test environment.</p> <p>Partner companies B2 to Bn develop software. Developed software is gathered by B1.</p> <p>Companies B2 to Bn cannot access the server inside B1 by security policy. Companies B2 to Bn are rather small and there is no CVS and GNATS server</p>	<p>Company C acts as group PM and develops software with group software company C1 and partner companies C2 to Cn in a unified structure.</p> <p>CVS, GNATS server and development environment are run by company C1.</p>	<p>Group software company D1 acts as group PM and develops software with partner companies D2 to Dn.</p> <p>CVS, GNATS server and development environment are run in D1 as group common environment.</p> <p>Company D1 cannot access company D's server by security policy.</p>	<p>Company E is separated into two divisions. Each division develops a part. Division Ea acts as PM of Ea group and develops software with partner companies Ea1 to Ean. CVS and GNATS server and development environment are run by Ea. Eb acts as PM of Eb group and develops with group software company Eb1 and partner companies Eb2 to Ebn. CVS, GNATS and development environment are run by Eb1.</p>	<p>Group software company F1 acts as group PM and develops with partner companies F2 to Fn. CVS, GNATS and development environment are run by companies F2 to Fn. There is no development environment in company F1.</p>

### 3. VERIFICATION OF USEFULNESS OF the MEASUREMENT

#### 3.1 Qualitative Estimation of Measurement

This kind of multivendor development structure generally conceals the development process for each IT vendor until system integration testing at the end of the development process. This project used measurement to reveal the development processes to the project owner and project manager. Measurement and analysis revealed development process differences and allowed better project management.

#### 3.2 Analysis Based on Post-Process Interview of Key Persons in the Project

The general features of the target project are similar to many government procurement projects in Japan. In particular, several

major IT vendors cooperated on the software development in a hierarchical structure with several work areas.

The following section evaluates project measurement and feedback based on interviews with key project personnel.

##### 3.2.1 Features of the structure of the target project

The consortium developed and managed the specification. METI provided development funding directly to each consortium company. While one company provided project management, human resource management and financial control were handled independently by each company.

In the consortium organization, six companies perform development. One of these six also was the project manager. A seventh company, the automobile company, played the role of system user.

Each development company developed their part of the system independently, providing developed parts to the integration test environment. Since there is no main contractor, access to development management information in other companies is limited. While requirements definition, basic design, and common interfaces in the detailed design are shared, most of the detailed design, program size, and source code are confidential.

### 3.2.2 Analysis of feedback usefulness

After successful completion of the project, key project personnel were interviewed concerning the usefulness of measurement and feedback. Table 2 shows the outline of these opinions.

There were about 60 persons engaged in this project. The interviews indicated the following eight positions in the project structure:

- 1) Project Owner
- 2) Person responsible for target system
- 3) General Project Manager for system development
- 4) Project leader for each subsystem (project management inside development companies)
- 5) Subsystem development leader who does not see development field. Trusted capabilities of lower organizations and left all management to them.
- 6) Subsystem development leader willing to see development field, but unable to do so. Insufficient time available to see work of individual development teams.
- 7) Subsystem development leader who could see development field, with resources to manage development teams.
- 8) Actual program development personnel

Table 2 indicates differing effects based on position. Opinions from interviews of key personnel support the usefulness of this project measurement and feedback approach. Selection and tailoring of feedback information to match positions and information needs is an important future issue identified in these interviews.

### 3.2.3 Interpretation of Results

The measurement process used in this project provided automated data collection across the members of the consortium and aggregation through a trusted third-party (SEC/EASE). Analysis results were then made available through a semi-automated feedback process involving customization and selection from the many analyses and graphics available. At least three factors strongly influenced the selection of information that was useful feedback.

First, the level of the individual strongly influenced the amount of aggregation or detail desired. The project owner, responsible person, and general project manager wanted bird's-eye views of the entire project. Project and subsystem leaders were more focused on their specific company or subsystem details, and at least in some cases, were not even interested in details from subsidiaries. Finally, the project development personnel typically

were most interested in details related to their own specific part of the work.

Second, related to this is the limited time and resources available to different individuals in the hierarchical structure to absorb and understand the analysis results. Even if additional details are available despite concerns for confidentiality and corporate privacy, often individuals do not have time to consider them. This emphasizes the need to provide the right information in feedback.

Third, the goals of the individuals differ significantly, which shapes the kind of information and analyses that are desired. For example, program development personnel are mostly interested in seeing how effective they are at their work. Subsystem development leaders and individual company project leaders are mostly interested in seeing the rate of completion and any trouble spots in their process. The highest levels again are focused on making the entire project across the different companies run smoothly and deliver.

The key finding of this study, then, concerns the need for customization and selection of feedback to match the various positions within the hierarchical structure of the software consortium. It is necessary to collaborate closely with the software development organizations to understand the roles and positions, and to tailor the delivery of feedback to best support those different parts of the organizations. Figure 4 shows this.

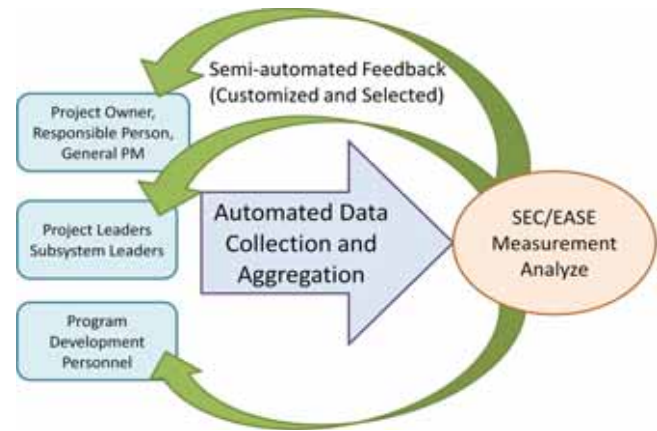


Fig.4 Data Collection, Analysis and Feedback Cycle

## 4. CONCLUSIONS

This research showed the effectiveness of in-process project measurement and feedback in a governmental consortium-type software project. The research suggests a new viewpoint on process improvement and innovation. Specifically, software development organizations may not be simple flat organizations composed of an owner, project manager, and development persons. Instead, they may have various ad hoc, complicated hierarchical structures. Efficient measurement and feedback mechanisms for projects must reflect this complexity of structure. The research was useful in obtaining a wide viewpoint concerning issues with measurement and feedback.

**Table2. Opinions of Project Individuals**

<b>Position</b>	<b>Opinions</b> o affirmative x negative + others	<b>Future Plans</b>
Project owner	o satisfied with project success o satisfied with development process	Planning next project Wishes to continue research
Person responsible for target system	o welcomed lack of software issues	Planning application of software
General Project Manager	o appreciated early feedback and visibility, esp. EPM x individual analysis tools immature, need development + has suggestion for feedback information	Wants to try on another new project Wishes to exploit pressure on software developers of visual presentation of their work
Project leader for each subsystem	o welcomed lack of software issues o had positive evaluation report about software engineering research from developers	Introduce tools and methods experienced in research into own company
Subsystem development leader who doesn't see development	o satisfied with project success o unable to evaluate software engineering research	Join another extended software engineering project with IPA/SEC
Subsystem development leader willing, but unable to see development	o information feedback useful for project management o project monitoring tools like EPM especially effective	Start new joint project with EASE
Subsystem development leader who could see development	o individual analysis tools more welcome than monitoring tools x some tools need development	
Actual project development personnel	o strongly supported visualization of development o positive effects on rewarding work	reported positive response to upper management

## 5. ACKNOWLEDGMENTS

This work is supported by IPA/SEC, METI and the MEXT of Japan. We thank researchers in the SEC and EASE & StagE projects who kindly support our project.

## 6. REFERENCES

- [1] Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Satoshi Iwamura, Mike Barker, Ken-ichi Matsumoto, An empirical trial of multi-dimensional in-process measurement and feedback on a governmental multi-vendor software project. *International Symposium on Empirical Software Engineering (ISESE) 2005, Vol.2*, Noosa Heads, Australia, Nov (2005) pp.5-8
- [2] Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Satoshi Iwamura, Yoshiki Higo, Katsuro Inoue, Mike Barker, Ken-ichi Matsumoto: Effect of Software Industry Structure on a Research Framework for Empirical Software Engineering, *International Conference on Software Engineering (ICSE)2006, Far East Experience Track, Poster Session*, Shanghai, China, May (2006) pp.616-619
- [3] Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Naoki Ohsugi, Akito Monden, Yoshiki Higo, Katsuro Inoue, Mike Barker, and Ken-ichi Matsumoto: A Proposal for Analysis and Prediction for Software Projects using Collaborative Filtering, In-Process Measurements and a Benchmarking, *International Conference on Software Process and Product Measurement (MENSURA)2006*, , Cadiz, Spain, November (2006) pp.98-107
- [4] Philip M Johnson: Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System. *International Symposium on Empirical Software Engineering and Measurement (ESEM) 2007*, Madrid, Spain, (2007), pp.81-90
- [5] M. Ciolkowski, J. Heidrich, F. Simon, M. Radicke: Empirical Results from Using Custom-Made Software Project Control Centers in Industrial Environments, *International Symposium on Empirical Software Engineering and Measurement (ESEM) 2008*, Kaiserslautern, Germany, (2008), pp.243-252
- [6] Alberto Colombo, Ernesto Damiani, Fulvio Frati, Sergio Oltolina, Karl Reed, Gabriele Ruffatti : The Use of a Meta-Model to Support Multi-Project Process Measurement, *Asia-Pacific Software Engineering Conferenc (APSEC) 2008*, Beijing, China, (2008), pp.503-510
- [7] N.Ohsugi, A.Monden, S.Morisaki: Collaborative Filtering Approach for Software Function Discovery: *International Symposium on Empirical Software Engineering (ISESE) 2002, vol.2*, Nara, Japan (2002) pp.45-46

# Prest: An Intelligent Software Metrics Extraction, Analysis and Defect Prediction Tool

Ekrem Kocagüneli<sup>1</sup>, Ayşe Tosun<sup>1</sup>, Ayşe Bener<sup>1</sup>, Burak Turhan<sup>2</sup>, Bora Çağlayan<sup>1</sup>  
<sup>1</sup>*Software Research Laboratory (Softlab), Computer Engineering Department, Boğaziçi University, Turkey*  
<sup>2</sup>*National Research Council (NRC), Canada*  
*ekrem.kocaguneli@boun.edu.tr, ayse.tosun@boun.edu.tr, bener@boun.edu.tr, Burak.Turhan@nrc-cnrc.gc.ca, bora.caglayan@boun.edu.tr*

## Abstract

Test managers use intelligent predictors to increase testing efficiency and to decide on when to stop testing. However, those predictors would be impractical to use in an industry setting, unless measurement and prediction processes are automated. Prest as an open source tool aims to address this problem. Compared to other open source prediction and analysis tools Prest is unique that it collects source code metrics and call graphs in 5 different programming languages, and performs learning based defect prediction and analysis. So far Prest in real life industry projects helped companies to achieve an average of 32% efficiency increase in testing effort.

## 1. Introduction

The role of software measurement becomes increasingly important to understand and control mature software development practices and products [1]. Software measurement helps to evaluate the software quality by measuring error-proneness of software modules, since residual defects in the software affects the final quality. However, measurement programs cannot be easily employed in software companies [2]. There has to be a *tool support* to analyze the quality of the software using various source code metrics [3,7]. Many researchers also have been working on building predictive models: defect prediction, cost/ effort estimation. These models need raw data (i.e. regular measurement of software attributes) [4,26,24,25,27,28]. These research have significant implications in practice as well. Predictive models support practitioners to take critical decisions under uncertainty. Automated tools help researchers and practitioners to measure software artifacts seamless to coders [4,24,25,26,27,28]. Current software

development environment, on the other hand, is complex such that multiple platforms (i.e. hardware and software) as well as multiple programming languages have to co-exist. Therefore any automated code measurement and analysis tool should address the issue of heterogeneity in software systems.

There exist several measurement and analysis tools, which are provided either as commercial of-the-shelf (COTS) [5, 22, 23] or as open source tools [6, 15, 19, 20, 21]. There are COTS tools, which provide extensive set of metrics and functionalities; however, they are not always affordable. Furthermore, their output formats cannot be easily integrated with other measurement and analysis tools. Open source tools, on the other hand, [6, 15] are easily accessible and their functionalities may be tailored to meet specific needs. However, open source tools have certain deficiencies: a) they can extract only a limited number of static code attributes from a limited number of programming languages, b) they do not include a learning based prediction support and c) they lack multiple output formats [8].

In this paper, we introduce an intelligent open source, software metrics extraction, analysis and defect prediction tool, called Prest [16]. The need for Prest has emerged during our collaborative research with industry partners from various domains (i.e. telecommunication [13], embedded systems [11] and healthcare [12]) over the past four years. Our aim in developing Prest was to extract static code attributes from software programs and build a learning-based defect predictor, which would highlight defect-prone parts of new projects, using code attributes and defect data from past projects. Prest is capable of extracting 28 static code attributes and generating call graphs by using five different language parsers. It also provides output in various formats that are compatible with popular toolkits like Weka [10]. Our industry partners have been using Prest for two years. The project

managers have been able to detect problems in coding practices and testing, and they take corrective actions on a timely manner.

## 2. Functionality

Prest is developed as a one-stop-shop tool that is basically capable of:

- ❖ Extracting common static code metrics from C, C++, Java, JSP and PL/SQL languages
- ❖ Presenting output via GUI components and in \*.xml, \*.csv, \*.xls and \*.arff file formats
- ❖ Generating call graphs in class and method level
- ❖ Defining new metrics or thresholds on extracted metrics
- ❖ Applying machine learning methods for analysis and defect prediction.

Each of these functionalities will be further described using a sample code (Figure 1). We also placed the sample code of Figure 1 and an executable jar of Prest in the Prest repository [16] for self trial. More complex analysis including defect prediction will be provided as a demo in Section 4.

```

public class Trial1 {
    public void trial1Func1()
    { trial1Func2(); }

    public void trial1Func2()
    { System.out.println("Trial1 class, function #2"); }

    public static void main(String args[])
    {
        Trial1 starter = new Trial1();
        Trial2 testClass = new Trial2();

        starter.trial1Func1();
        testClass.trial2Func2();
    }
}

public class Trial2
{
    public void trial2Func1()
    { System.out.println("Trial2 class, function #1"); }

    public void trial2Func2()
    { trial2Func1(); }
}

```

Figure 1. Sample code

### 2.1. Parsing and saving a project

Prest can parse all files that are written in different programming languages by using different parsers at the same time. Similar tools, on the other hand, can parse only one language at a time while ignoring other files. Once a project, such as the sample code in Figure 1, is parsed, the metrics are presented via GUI components in a structured manner and outputs are placed under the related project folder within the repository. The outputs are presented in several formats: \*.csv, \*.xls, \*.arff and \*.xml. In Figure 2, only

one metric, i.e. cyclomatic density, is presented as the static code attributes that can be extracted by Prest, since we have page limitations. However, Table 1 provides full set of attributes.

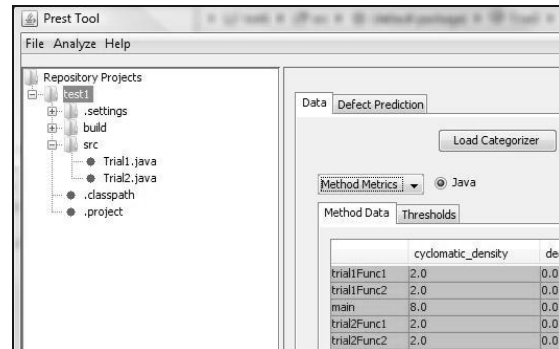


Figure 2. GUI Overview of Prest

### 2.2. Call graph generation

Prest introduces a new and simple call graph feature for all supported languages. It extracts this information to better illustrate dependencies between functions/classes and the complexity of software systems. Basically, a function call graph represents the encodings of caller-callee relations between functions in a structured manner (Figure 3). Using Prest, each function in Figure 3 is treated as a potential caller and a unique ID is assigned to each function. Therefore, all the functions are listed under the column CALLER\_NAME and their ID's are listed under the column CALLER\_ID in an excel file. The second CALLEE\_ID column keeps the ID's of the called function(s) that were called by the caller function. Generated call graph matrix of Prest can be seen in Figure 4.

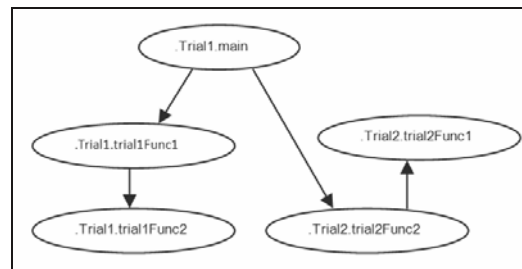


Figure 3. Function calls of sample code

CALLER NAME	CALLER ID	CALLEE ID
.Trial1.trial1Func1	3	4
.Trial1.trial1Func2	4	
.Trial1.main	5	3 10
.Trial2.trial2Func2	10	9
.Trial2.trial2Func1	9	

Figure 4. Call graph matrix of sample code

### 2.3. Data Analysis and Prediction

Data analysis and prediction are particular features of Prest, which provide analysis and prediction via Naïve Bayes (NB) and Decision Tree (DT) algorithms. Given actual defect data of a project, in which bugs are matched with functions; Prest can analyze the given data via Naïve Bayes and Decision Tree algorithms and make predictions for a future release, which has yet not been tested. Then, it pinpoints defect-prone modules to increase the testing efficiency considerably. This feature has drastically helped our industry partners by reducing the testing effort by 32% [11]. Its architecture and a detailed tutorial will be explained in Section 3, 4.

### 2.4. Threshold and New Metric Definition

Certain values of metrics or a combination of those may be indicator of error proneness. Prest provides users the ability to define certain conditions (thresholds) on the extracted metrics and apply color coding according to user-defined thresholds, i.e. metrics of defect-prone modules are colored with red on the GUI, whereas the defect-free ones are painted as green. Furthermore, Prest lets users to define new metrics by combining existing metrics via mathematical operators. In Figure 5, definition of a new metric using “/ DIVIDE” operator, cyclomatic\_complexity and lines\_of\_code metrics is illustrated.

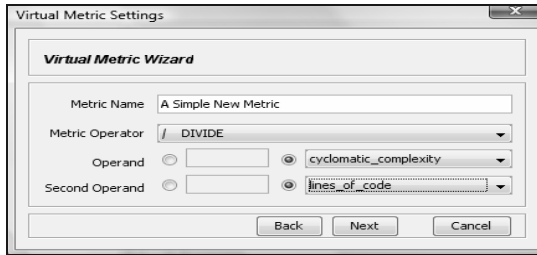


Figure 5. Defining a new metric

## 3. System Architecture

Prest architecture has four main components: Language parser, metric extractor, analysis and prediction component and GUI components.

### 3.1. Language Parser

A parser is responsible for parsing code into tokens depending on its type such as operand, operator etc. Currently, Prest consists of C, C++, Java, JSP and PL/SQL parsers.

### 3.2. Metrics Extractor

Once the language parser is done with parsing the code into tokens, the metric extraction component starts to execute and it produces logical results depending on the output of the language parser. Those logical results are used to calculate static code metrics, listed in Table 1. Prest collects 28 static code attributes (Table 1) and none of the other open source metrics extraction tools [18] were able to extract all metrics.

Table 1. Static code metrics extracted by Prest

Total loc	Blank LOC
Comment LOC	Code Comment LOC
Executable LOC	Unique Operands
Total Operands	Total Operators
Halstead Vocabulary	Halstead Length
Halstead Volume	Halsted Level
Halstead Difficulty	Halstead Effort
Halstead Error	Halstead Time
Branch Count	Decision Count
Call Pairs	Condition Count
Multiple Condition Count	Cyclomatic Density
Cyclomatic Complexity	Decision Density
Design Complexity	Design Density
Normalized Cyclomatic Complexity	Formal Parameters

### 3.3. Analysis and Prediction Component

Analysis and prediction component of Prest significantly differentiates itself from similar open source tools, since none of them provides a learning based defect prediction component [18]. Unlike other open source metric extraction tools, Prest can perform analysis and predictions regarding the defect-proneness of software by utilizing machine learning methods. We have benefited from Weka libraries [10] to implement two classifiers, Naïve Bayes and Decision Tree, for this component. However, new methods may be included either by implementing it from scratch or by calling Weka libraries.

### 3.4. GUI Component

GUI component is responsible for interacting with the user and presenting the results. We paid particular



attention to GUI component and analyzed various tools such as Eclipse [6], Predictive [5] and WEKA [10], before designing it. We aimed to keep the usage simple, while providing full range of features, such as project repository, easy switch between metric extraction and analysis tabs, defining thresholds on static code metrics, filtering results depending on defined thresholds, applying color codes, and defining new metrics.

#### 4. Demo

In Section 2, we have analyzed a sample Java code to discover the functionalities of Prest. In this section, we have analyzed a large software system from one of our industry partners. This software system has been implemented in Java and JSP languages. We took two versions of the same system (version 11 for training and 12 for testing) and extracted static code attributes from both Java and JSP files with Prest. Then, we matched actual defect data with the files whose static code attributes were extracted by Prest and fed them to analysis and prediction component. We have used Naïve Bayes classifier to predict defect-prone files in version 12. Finally, we have measured the performance of the prediction component of Prest when only Java files, only JSP files and both of them are used. In Figure 6, probability of detection rates (pd) and the balance rate (bal) have been increased when both Java and JSP files are used. Furthermore, probability of false alarm rates (pf) has been decreased.

Those results have been encouraging in the sense that extracting static code attributes from all the languages of a software project can increase the prediction performance. In addition, Prest, as a single tool, is able to conduct a thorough analysis in large software systems, thereby reducing the need for multiple tools for different languages and machine learning tools.

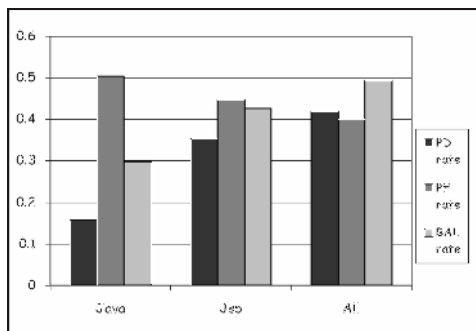


Figure 6. Improvements in the prediction performance of Prest

#### 5. Development Methodology

Prest has been developed by MS and PhD students in SoftLab during the last three years. At various lengths of involvement (from 6 months to three years), a total of 12 students and a faculty member worked as the developers and designers of Prest. We used a formal waterfall approach where we took the requirements from our industry partners, reviewed them and used existing tools. Then, we designed the architecture, coded Prest, and conducted alpha and beta tests with our industry partners. Also, a senior architect has been guiding us for the current and future architecture of the tool. All development stages are well documented and we have used a versioning system as well as an automated bug tracking system. Current members of SoftLab carry out implementation of new parsers and they provide maintenance of Prest.

#### 6. Current Usage & Benefits

Early versions of Prest were used by a local white-goods manufacturer, who wanted to measure code quality to reduce defect rates and to effectively manage their testing resources [11]. Using Prest, we collected static code metrics attributes from C codes at function level. Then, we analyzed the defect-prone parts of the software using data analysis component of Prest and found that testing effort could be reduced by 32% while catching 76% of defective modules [11].

Recently, we have conducted a metrics program in a large telecommunication software system [13]. In this project, we collected static code metrics with Prest in Java source file-level. Then, we matched those files with actual defect data and used Naïve Bayes classifier to predict defective files of the software. We have also used call graph information in method level and applied the Naïve Bayes classifier to predict defect-prone files in the system. Results show that prediction model in Prest has been capable of detecting 84% of defective files by inspecting only 31% of the code [13].

In addition to our local industry partners, currently Prest has been in use in a multi-national company in the UK. Since Prest [16] is designed as an open source tool, it is available via Google Code [19] to review, download or further develop and integrate.

#### 7. Support

The development team of Prest provides support to users [14]. Once a development activity is performed and a stable version is elicited new code is committed to the Prest repository in Google Code [16]. Therefore,

the code that users can access is always the latest stable version of Prest. Any failure or problem in the system can be directly entered into the issue management system of Google Code in order to track the status of each problem on the web.

## 8. Related Work

There exist a considerable number of software metrics tools available either as open source [6, 15, 19, 20, 21] or as commercial [5, 22, 23]. Since Prest is developed as an open source tool, we focus on non-commercial tools for comparison of Prest and other tools. We acknowledge that there is no ultimate criterion to compare different tools and conclude that one is certainly better than the other. However, a set of criteria may be defined while assessing different metric tools: Number of languages that are supported, number and nature of metrics extracted, type of output formats and analysis and prediction components. Those functionalities are also examined by other researchers [7, 18]. Thus, they are also critical for our future extensions in Prest. We have presented this comparison with CCCC [19], Chidamber-Kemerer Java Metrics [20], Dependency Finder [21], Eclipse Metrics Plug-in version 1.3.6 [6] and CyVis[15] tools in Table 2. From

Table 2, we can see that Prest is more extensive than other open source tools with respect to languages it parses, number of extracted metrics, output formats and its analysis and prediction component. However, this does not make Prest the finest and the ultimate tool, since there has been significant effort behind each tool and we still lack some properties such as simple and precise graphical representation of dependencies in Eclipse plug-in or saving extracted metrics in an html file. Nevertheless, we have managed to provide an all-in-one tool for software practitioners by saving their time and effort for searching multiple tools for various needs and dealing with various output formats. Moreover, we have benefitted from Prest in our research studies by extracting static code attributes and doing predictions for all experiment settings.

## 9. Conclusion and Future Work

Prest has been in use in three large software systems (locally and internationally). It has also been used in various SoftLab empirical research studies at different companies [11, 12, 13]. Prest in practice with its prediction capability has so far successfully guided project managers to take decisions under uncertainty and has considerably increased testing efficiency.

**Table 2. Comparison of Prest and other open source tools**

		Prest	CCCC	CK Java Metrics	Dependency Finder	Eclipse Plug-in	CyVis
<b>Supported Languages</b>	C	+	+	-	-	-	-
	C++	+	+	-	-	-	-
	Java	+	+	+	+	+	+
	Jsp	+	-	-	-	-	-
	PL/SQL	+	-	-	-	-	-
<b>Output</b>	csv	+	-	-	-	-	-
	xls	+	-	+	-	-	-
	arff	+	-	-	-	-	-
	xml	+	+	+	+	+	+
	html	-	+	-	+	+	-
<b>Data Analysis Component</b>	+	-	-	-	-	-	-
<b>Call Graph Generation</b>	+	-	-	-	+	+	-
<b># Metrics Collected</b>	28	9	6	13	23	2	

Going forward, Prest will be constantly adding new parsers as well as more learning algorithms. Currently, we are in the process of migrating Prest tool to cloud computing in order to serve larger communities better, to share data and foster reproduction of empirical experiments.

## 10. Acknowledgment

This research is funded in part by Tubitak EEEAG108E014. We would also extend our gratitude to Mr. Turgay Aytaç, senior architect, for his guidance as well as A.D.Oral, E.G. Isık, C. Gebi, H. Izmirlioglu, O. Bozcan and S. Karagulle for their efforts in the development of this tool.

## 11. References

- [1] B. Kitchenham, S. L. Pfleeger, and N. Fenton. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 21(12):929–944, 1995.
- [2] N. Fenton. Software measurement: a necessary scientific basis. *Software Engineering*, *IEEE Transactions on*, 20(3):199–206, Mar 1994.
- [3] M. J. Harrold. Testing: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 61–72, New York, NY, USA, 2000. ACM.
- [4] N. Nagappan and T. Ball. Static analysis tools as early indicators of pre-release defect density. *Software Engineering*, 2005. *ICSE 2005. Proceedings. 27th International Conference on*, pages 580–586, May 2005.
- [5] Predictive, Integrated Software Metrics, available at <http://freedownloads.rbytes.net/cat/development/other4/predictive-lite/>
- [6] Eclipse metrics plug-in 1.3.6, available at <http://sourceforge.net/projects/metrics>.
- [7] P. Kulik and C. Weber. Software metrics best practices – 2001. In *Software Metrics Best Practices 2001*, March 2001.
- [8] M. Auer, B. Graser, and S. Biffl. A survey on the fitness of commercial software metric tools for service in heterogeneous environments: common pitfalls. *Software Metrics Symposium*, 2003.
- [9] B.Turhan, G. Kocak and A. Bener. Software Defect Prediction Using Call Graph Based Ranking (CGBR) Framework, *Proceedings of the 34th EUROMICRO Software Engineering and Advanced Applications (EUROMICRO SEAA'08)*, 2008.
- [10] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.
- [11] A. Tosun, B. Turhan and A. Bener. Ensemble of Software Defect Predictors: A Case Study. *Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM'08 Short Paper)*, pp.318-320, 2008
- [12] A. Tosun, B. Turhan and A. Bener. The Benefits of a Software Quality Improvement Project in a Medical Software Company: A Before and After Comparison, *International Symposium on Health Informatics and Bioinformatics (HIBIT'08 Invited Paper)*, 2008.
- [13] A. Tosun, B. Turhan and A. Bener. Direct and Indirect Effects of Software Defect Predictors on Development Lifecycle: An Industrial Case Study, to appear in *Proceedings of the 19th International Symposium on Software Reliability Engineering (Industry Track)*, 2008.
- [14] Software Research Laboratory (Softlab), available at [www.softlab.boun.edu.tr](http://www.softlab.boun.edu.tr)
- [15] Cyvis Software Complexity Visualizer, <http://cyvis.sourceforge.net/>
- [16] Prest Metrics Extraction and Analysis Tool, available at <http://code.google.com/p/prest/>.
- [17] T. Menzies, J. Greenwald and A. Frank. Data Mining Static Code Attributes to Learn Defect Predictors, *IEEE Transactions on Software Engineering*, January 2007, Vol.33, No. 1, pp. 2-13.
- [18] R. Lincke, J. Lundberg, W. Löwe. Comparing Software Metrics Tools, *ISSTA '08: Proceedings of the 2008 international symposium on Software testing and analysis*, 2008
- [19] C and C++ Code Counter, available at [sourceforge.net/projects/cccc](http://sourceforge.net/projects/cccc), 2006.
- [20] D. Spinellis. Chidamber and Kemerer Java Metrics, available at [www.spinellis.gr/sw/ckjm](http://www.spinellis.gr/sw/ckjm), 2006.
- [21] Dependency Finder, available at [depfind.sourceforge.net](http://depfind.sourceforge.net), 2008.
- [22] Analyst4j Find Using Metrics, available at [www.codeswat.com](http://www.codeswat.com).
- [23] SciTools Source Code Analysis and Metrics, Understand for Java, available at [www.scitools.com](http://www.scitools.com)
- [24] Victor R. Basili , Lionel C. Briand , Walcélio L. Melo, A Validation of Object-Oriented Design Metrics as Quality Indicators, *IEEE Transactions on Software Engineering*, v.22 n.10, p.751-761, October 1996
- [25] S. R. Chidamber , C. F. Kemerer, A Metrics Suite for Object Oriented Design, *IEEE Transactions on Software Engineering*, v.20 n.6, p.476-493, June 1994
- [26] Nachiappan Nagappan , Laurie Williams , John Hudepohl , Will Snipes , Mladen Vouk, Preliminary Results On Using Static Analysis Tools For Software Inspection, *Proceedings of the 15th International Symposium on Software Reliability Engineering*, p.429-439, November 02-05, 2004
- [27] Yue Jiang , Bojan Cuki , Tim Menzies , Nick Bartlow, Comparing design and code metrics for software quality prediction, *Proceedings of the 4th international workshop on Predictor models in software engineering*, May 12-13, 2008, Leipzig, Germany
- [28] A. Gunes Koru , Hongfang Liu, Building Defect Prediction Models in Practice, *IEEE Software*, v.22 n.6, p.23-29, November 2005

## Accelerated Risk Management using Statistical Triggers

**Rose Williams, M.S.C.S.**

Department of Software and Services Science  
I.B.M. Thomas J. Watson Research Center  
Hawthorne, N.Y. 10532, U.S.A.

[rosemw@us.ibm.com](mailto:rosemw@us.ibm.com)

**Krishna Ratakonda, PhD.**

Department of Software and Services Science  
I.B.M. Thomas J. Watson Research Center  
Hawthorne, N.Y. 10532, U.S.A.

[ratakond@us.ibm.com](mailto:ratakond@us.ibm.com)

### **Abstract**

*In any large IT services provider's portfolio, it is not uncommon to find several deals that have a high customer satisfaction rating, but result in a financial loss. In many cases it is not clear that an IT engagement that is in serious trouble should be immediately terminated. The provider may have good reasons in continuing an engagement which may not be eventually profitable. For example, the provider wants to maintain a relationship with a client, or avoid an adverse impact on their reputation. It is important to not only identify that a project has become troubled, but also be able to predict whether the project can be salvaged from financial, quality or other perspectives. By drawing upon a historical database of services projects spanning several years, we are able to draw conclusions on the effectiveness of certain well established principles in risk assessment used by the project management community. We define derived statistical measures that can be used to predict the eventual outcome of an in-delivery project along several dimensions. In this paper, we will explore how quantifiable measures of project progress, gathered at several important stages of a project's life-cycle, can aid in early identification of troubled projects.*

### **1. INTRODUCTION**

What causes complex IT services engagements to fail? Patterns of failure that emerge can be associated with a number of root causes, many of which are well known, such as: requirements creep, sudden changes in scope/schedule changes or poor project management practices. Some of these root causes are particularly difficult to address in large engagements because of the complex interactions between the different portions of a project which are typically addressed by separate teams that do not communicate on a regular basis. The large, diverse and dynamic nature of the global workforce that many IT providers use to deliver on these engagements has put a new level of stress on traditional practices. Process guidance only goes so far when the team responsible for delivery has limited experience using the processes in other engagements. Resource churn and

communication gaps play a significant role in undermining what would otherwise be effective risk mitigation measures. IT service leaders have taken significant steps to address these challenges; such measures range from organizing global resource pools into cohesive competency centers and instituting common tooling platforms across the board. However, quantifying the overhead and risks involved in engaging a geographically distributed team to deliver on an IT engagement is still more of an art than a science.

Given a collated historical database of project metrics and outcomes, statistical analysis can be used to derive classification rules, regression measures and identify statistically significant clusters in the dataset. Our system analyzes the quantitative metrics collected at different stages of an IT engagement, including risk management reviews, technical reviews and monthly financial reports, to gain an early insight into key patterns of trouble. As such, statistical measures can be used to initiate accelerated risk mitigation plans early in the project's lifecycle resulting in a reduction of potential loss. Our predictive analytics detect troubled patterns early in the project's life-cycle, and act as first-alerts helping focus the limited risk management resources on the subset of the projects that are most at risk.

### **2. LITERATURE SURVEY**

In recent literature, the trend towards a systems approach to risk management, wherein statistical techniques are used to classify and mine the project metrics data, is gaining acceptance [1, 2, 3, 4]. These methods actively use the metrics collected during traditional risk management reviews and then employ techniques borrowed from statistical learning theory to derive models that describe the relationship between the collected metrics and eventual project outcomes.

Quantitative analysis of IT investment decisions using options analysis, to mitigate *financial risk*, is an active area of research [5, 6, 7]. Options analysis can be used to assess the value of prototyping work and early adoption initiatives related to new IT platforms

– options provide a way to evaluate the value of IT projects which give the *right* to adopt the resulting technology without having the *obligation* to do so. Fichman [5] shows how options analysis can be used to predict IT platform initiation and adoption, value IT platform options and manage IT platform implementation. Chen and Sheng [6] discuss how to do real options analysis while accounting for estimation errors.

In our view, the lack of widespread adoption for options analysis can be traced back to a fundamental issue that affects all methods for analyzing *financial risk* – the inability to accurately estimate with any degree of certainty the net present value (NPV) of a complex in-flight IT project. In this paper, we evaluate financial risk by analyzing how projects that exhibited similar trends in project metrics performed in the past – thus, the analysis has no dependence on a particular methodology for evaluating NPV.

Qualitative analyses of IT projects remain an active area of research. Erickson and Evaristo [8] provide an account of how risk factors associated with IT projects are magnified or multiplied when dealing with distributed project teams. They provide a conceptual list of a variety of factors ranging from culture to distance and discuss how an increase in *distributedness* along these dimensions affects project risk. Ramasubbu and Balan [9] present an empirical study which quantifies the loss in quality and increase in schedule risk that one would expect in global software projects. They suggest that good software process controls may mitigate these losses to some degree. Beise [10] suggests that good project management practices when properly applied may help to mitigate some of the problems caused by virtual teams. Our analysis suggests that even under the best possible control structure the losses due to distributed development are unavoidable – the ability to recognize the symptoms and act quickly will decide which companies will succeed in leveraging the promise of globalization.

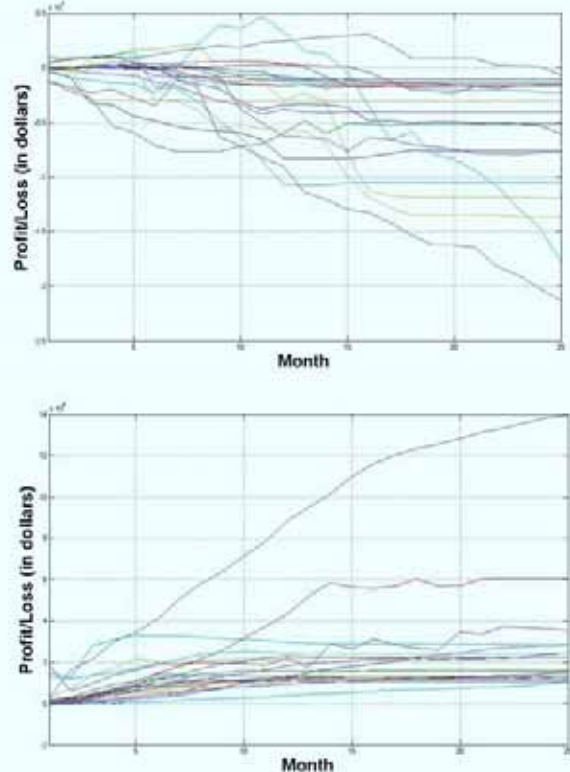
### 3. MINING IT PROJECT METRICS FOR TROUBLE PATTERNS

In this section we will take a closer look at patterns in project metrics that are indicative of future trouble. As discussed in the previous sections, the risk entailed can be along several dimensions – financial, functionality, customer satisfaction or others. In section 3.1, the focus is on financial risk for the delivery organization with a view to determining whether a given project will be eventually profitable at completion. In section 3.2, we discuss how to

predict a poor outcome in an aggregated project risk measure, in the preliminary phases of an IT application services project, based on information gathered prior to project inception. Section 3.3 shows how this prediction algorithm can be extended to an in-flight project where there is an established history of past risk assessment reviews. In particular, the emphasis is on understanding what role early patterns in individual dimensions of project risk play in determining a poor outcome in an aggregated project measure in the future.

#### 3.1 Financial Metrics

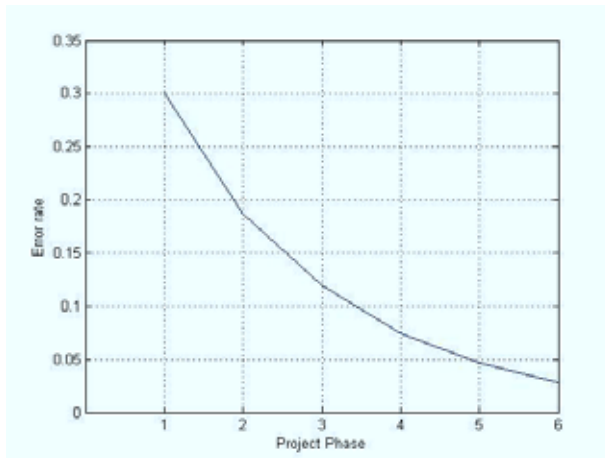
In analyzing project financial data, our objective is to determine whether a project will be eventually profitable. In our experience, we found that a small set of projects were the cause for a significant drop in profits. Therefore, if we could identify these projects early in the lifecycle, there is large potential for increasing the profit margins for the rest of the projects – terminating a fraction of these high loss projects early could increase the margin by about 10-20%.



**Figure 1:** A comparison of profit data plotted on a monthly basis from projects that resulted in a loss (top) and those that resulted in a profit (directly above). Monthly data is plotted only for the first 25 months from project start date.

Figure 1 shows the profiles of the project project/loss over a project's lifecycle. It is clear that most profitable (loss making) projects start generating profits (losses) early on in their lifecycle. The graphs clearly show that the longer a project is unprofitable, the more difficult it is to turn-around. Our approach is to turn this *rule of thumb* into a statistically robust measure. One would expect that some projects are more front loaded than others i.e., significant ramp up costs are offset by realization of significant revenues in the later stages of project execution. However, this effect is reduced as we move further in time in the delivery phase.

More formally, given the monthly ledger data (revenue and cost) and static categorical information about a project, we would like to identify projects that will generate a loss at completion. Applying a classification algorithm, we identify the set of projects that will produce a significant loss. As expected, the prediction accuracy (in terms of both false positives and false negatives) improves as we move closer to project completion. Figure 2 shows a plot of the error rate on the training data for completed projects as a function of the project duration. To account for the variations in project duration, we normalized the project's lifecycle into six phases.



**Figure 2:** A plot of the error in classification for the training data set of completed projects as a function of the normalized project duration i.e., a project is classified based on partial data.

### 3.2 Qualitative Metrics

**Pre-engagement metrics:** A number of risk assessment related activities are conducted prior to project inception with an objective of determining the

risk entailed in delivery. The number and depth of these reviews is determined by a variety of factors, such as: the size of the project, the novelty of the proposed solution, and the industry sector to which the client belongs. These reviews may cross multiple delivery organizations within and beyond the purview of the primary service provider. The integrated technical review (ITR) is meant to integrate inputs from these different organizations and produce a proposal baseline assessment (PBA). Key decisions facilitated by this process include the amount of contingency budget that is allocated to the project, and the frequency and composition of project management reviews (PMR) during the project lifecycle. In the rest of this section, we will explore how to take advantage of this information to predict the outcome of the first project management review after project inception (the initial PMR or IPMR) which is a letter grade: A (best), B, C or D (worst) assigned within the first 12 weeks. The letter grades indicate increasing risk and/or project trouble.

**Data collection:** By reviewing the PBAs of several recent IT delivery projects, we identified a set of sixteen questions that were considered to be predictive of the project outcome in the initial phases of the project lifecycle. These questions were reviewed by experienced risk managers and carefully screened to ensure that they can be used to obtain objective answers within well-defined ranges. For each of the reviews used in this paper, a risk manager was involved in assessing the project and answering the questions. The intent was to remove subjectivity in the answers by basing them on data that is readily accessible to the risk manager. Questions ranged from a client's past experience with the delivery organization, match of the delivery team's technical skills to the project objectives and the type of contract (fixed price vs. time and materials). It is to be noted that these questions do not delve deeply into the technical details of the project itself, as they need to cover a variety of IT projects ranging from implementing a custom application to customizing a packaged application.

**Statistical Process:** We then used statistical classification algorithms to match the risk management questions with the eventual project outcome. Prior to classification, the answers to the risk management review questions were scaled and binned appropriately to reduce the effects of variance due to human error and differences in procedures followed in different countries. We chose to apply a decision tree classifier to the data set. This approach produced acceptable results and the resultant rules enable the end users to understand the prediction

process more readily. Using random sampling, we split the available data set into a training and test data set.

A cost matrix that indicated the relative weight to be associated with a particular type of misclassification was used. The cost matrix allows the risk managers to indicate the relative weight to be associated with a particular type of misclassification. In our model, misclassifying a project that would have been a C as a D project carries a cost (or weight) of 1, where as a more serious misclassification of C as an A carries a weight of 9. In our prediction model, false positives (good projects being classified as troubled) carry less weight than false negatives (troubled projects being classified as good).

**In-Delivery metrics:** Once a project is in delivery, there are periodic project management reviews (PMRs), conducted by experienced risk managers, the frequency of which are typically determined by the results of prior PMRs. The question we would like to answer in this section is whether an accelerated risk management review is warranted based on the data that is collected during the PMR process. Thus, the focus is on predicting the transitions of good projects (grade A or B) to troubled projects (grade C or D). The objective data collected during a PMR process can be categorized into three classes: static project characteristics, standard project management metrics, and root causes of troubles (ex: root causes can range from statement of work issues in design phase to resource or client issues in delivery phase).

Given this data, we trained a decision tree classifier to predict the next PMR score, as we did in the case of pre-engagement metrics. Given the complexity of the data set, we faced challenges in binning and scaling the data to make it amenable for a tree classifier. Also, the practices followed in conducting PMRs underwent changes over the years. Thus, we had to choose a narrow timeframe for the training data to avoid contaminating it with outliers. Figure 3 shows the transitions between two consecutive PMRs that occurred in the training data set. Applying the decision tree classifier on a test data set shows an ability to capture about 80% of the transitions with 14% false positives and 2% false negatives over the entire data set.

		Current PMR			
		A	B	C	D
Previous PMR	A	6	7	4	1
	B	7	412	102	52
	C	1	120	344	122
	D	1	50	73	521

Figure 3: Transitions between PMR types for consecutive PMRs conducted on the same project. The highlighted cells (in dark blue) show transitions of interest from a risk management perspective.

Although the analysis for predicting transitions is similar to the one used for pre-engagement metrics, the prediction findings in this case are used to cull out those projects that require closer attention from the risk management practice. As such, keeping a low false negative rate is important in this case to avoid missing any potentially negative transitions. Given that the algorithm can potentially miss 20% of the transitions, it is important to use this as a supplement and not as a replacement for established risk management practices. As one would expect, many of the transitions in PMR rating occur more frequently in the initial phases of the project – thus, conducting the first few PMRs on a regular basis is recommended irrespective of the predicted outcome.

#### 4. RECOMMENDATIONS

In implementing the algorithms and rolling them out for use in a large organization, we found that getting access to reliable, consistent data that can be statistically analyzed is typically the most challenging aspect of the project. IT services delivery organizations that are spread across the globe have different auditing and data collection requirements that make this process difficult. Based on our experience, we recommend the following best practices:

- Do a preliminary analysis of the data to establish a viable sub-set of attributes that have stable reporting characteristics.
  - Cross-check with sources for validity. It is important to access the data from primary/trusted sources. In large organizations the same data may be reported through multiple databases – and accessing data from a secondary source may lead to inconsistencies or stale data.

- Ensure that the data has attributes that are amenable to analysis. Some data is typically more prone to subjective interpretation due to its nature.
  - It is also key to ensure that the data collection process is itself not compromised by conflicts of interest. Typically, project health data entered by project managers, who have a vested interest in the success of the project, is not as reliable as reviews conducted by dedicated risk managers. Accountability or, at least, traceability between the data collected and the person responsible for entering the data is an important indicator of its trustworthiness.
- Identify a set of algorithms that are suitable for use. Our focus was on identifying portfolio wide-trends that can be aggregated (rolled up) meaningfully. Models that have easily accessible semantic interpretations (such as decision tree classification) are more suitable.
  - Perform sanity checks before introducing new metrics and data items
    - If new metrics need to be added, following a goal-question-metric or similar process to validate their need is a good practice. It is also important to perform statistical tests to validate that the data collected for each metric is valid – to expose problems in wording the question or the ability of the person to answer the question.
  - Choose the right set of attributes as inputs and outputs for analysis
    - Predictive analysis has more value (i.e., a higher predictive power) when the input attributes combine data that is collected early in a project's lifecycle and then periodically thereafter.
    - Target attributes should be carefully selected to match typical information that is relevant to delivery excellence: customer satisfaction, classification score, financial viability at completion etc.

## 5. CONCLUSIONS

In this paper, we considered the problem of generating early warning indicators that can be then used to perform proactive/accelerated risk management. Although it is not always possible to salvage a project, terminate a project or take other drastic measures, predictive algorithms can act as a first alert helping focus the limited risk management resources on the subset of the projects that are most

at risk. By analyzing a variety of IT application delivery projects over the span of a number of years, we trained and validated the prediction algorithms. The results are encouraging and show the potential for a significant impact in a variety of project planning areas.

## 6. ACKNOWLEDGEMENTS

We would like to thank Paul Huang for providing the data and algorithms used in section 3.1. We would like to thank Jack Bisceglia and Russell W. Taylor for their help in understanding the prevalent risk management practices.

## 7. REFERENCES

- [1] Sun, A. & Li, C. (2007), Research on Project Risk Evaluation Method Based on Markov Process, in 'Proc. International Conference on Wireless Communications, Networking and Mobile Computing WiCom 2007', pp. 5293--5296.
- [2] Hu, Y.; Huang, J.; Chen, J.; Liu, M. & Xie, K. (2007), Software Project Risk Management Modeling with Neural Network and Support Vector Machine Approaches, in 'Proc. Third International Conference on Natural Computation ICNC 2007', pp. 358--362.
- [3] Jianyi, G.; Li, Z.; Xusheng, L.; Yuejuan, H. & Zhengtao, Y. (2008), Implementing a quantitative-based methodology for project risk assessment DSS, in 'Proc. 27th Chinese Control Conference CCC 2008', pp. 730-734.
- [4] Clemons, E. & Weber, B. (Fall 1990), 'Strategic information technology investments: guidelines for decision making', *Journal of Management Information Systems* 7(2), 9-28.
- [5] Fichman, R. G. (2004) "Real Options and IT Platform Adoption: Implications for Theory and Practice", *Information Systems Research*, 15(2) 132-154.
- [6] Chen, L.; Sheng, O.; Goreham, D. & Watanabe, J. (2005), A real option analysis approach to evaluating digital government investment, in 'Proceedings of the 2005 national conference on Digital government research', Digital Government Society of North America, pp. 157-166.
- [7] Kumar, R. (Summer 1996), 'A note on project risk and option values of investments in information technologies', *Journal of Management Information Systems* 13(1), 187-193.



- [8] Erickson, J. M. & Evaristo, R. (2006), Risk Factors in Distributed Projects, *in* 'Proc. 39th Annual Hawaii International Conference on System Sciences HICSS '06', pp. 216c--216c.
- [9] Ramasubbu, N. & Balan, R. K. (2007), Globally distributed software development project performance: an empirical analysis, *in* 'ESEC-FSE '07: Proceedings of the 6th joint meeting of the European software engineering conference', ACM, New York, NY, USA, pp. 125-134.
- [10] Beise, C. M. (2004), IT project management and virtual teams, *in* 'SIGMIS CPR '04: Proceedings of the 2004 SIGMIS conference on Computer personnel research', ACM, New York, NY, USA, pp. 129-133.

# A Layered Approach for Planning Releases under Uncertain Capacities

Jim Mc Elroy & Guenther Ruhe

Laboratory for Software Engineering Decision Support, University of Calgary  
2500 University Drive NW, Calgary, AB T2N 1N4, Canada  
{mcelroy, [ruhe](mailto:ruhe@cpsc.ucalgary.ca)}@cpsc.ucalgary.ca

## Abstract

*Planning for product releases includes a number of uncertainties. This paper studies a layered approach for handling uncertainties related to available and the requested resources. The main idea of the approach is to solve a sequence of problems starting from most restricted to less resource restricted problems. By keeping the assignments of features of the former problem fixed for the solution at the next layer, we maintain compatibility of the solutions. The results of the approach is a set of feature release strategies. The important implication of the compatibility is that the different layers also provide guidance for in which order features should be treated.*

*A hypothetical case study is undertaken to illustrate the approach and to show the added value from a decision-making perspective.*

**Keywords:** *Software engineering decision support, release planning, resource planning, uncertainty.*

## 1 Motivation and Related Work

Release planning for incremental software development facilitates the optimal assignment of features to releases such that most important technical, resource and budget constraints are met. The notion of release planning varies from informal approaches, including those performed in agile development, to more formalized approaches. A discussion of the two fundamental approaches and existing solution methods is given in [16]. Informal planning approaches as described in [2] mainly rely on communication and expert knowledge. Agile release planning (often called “planning games”) falls into the informal category due to the lack of a formal optimization model defining it.

Formalized approaches provide an explicit and quantitative description of the problem to be solved. Based on that, mathematical optimization is applicable to generate optimal or approximate solution alternatives. This approach is used as the kernel of an evolutionary problem solving method called EVOVLE\* [15], the formal approach the authors have the most experience with. Other approaches with underlying formal models include those proposed by

Jung [11], Bagnall [1], Carlshamre [5], and van den Akker, et. al. [17].

One problem with formal approaches such as EVOLVE\* has to do with the accuracy of the inputs to the model. Effort estimation input, which serves as an important constraint on the model, is especially problematic. Effort estimations may be significantly off in the early stages of project planning, when release planning is typically done.

Therefore, the challenge addressed by this paper is how to provide a release planning strategy, which is implementable also under varying levels of availability of resources. The paper is divided into six sections. Section 2 gives a description of the formal EVOLVE\* approach to release planning. Section 3 discusses, in general, how to handle resource estimation error in the formal model. Section 4 delineates a layered approach addressing uncertainty in the availability of resources. Section 5 describes a case study using the specific approach. Finally, in section 6 we discuss the results and provide an outlook for future research.

## 2 Release Planning with EVOLVE\*

### 2.1 Features and Related Decision Variables

This paper uses the concept of a “feature” as the basic unit for release planning. Features are the “selling units” provided to the customer. In the context of this research, we follow the definition given by [18] which defines features to be “a logical unit of behavior that is specified by a set of functional and quality requirements”.

We assume a set of features  $F = \{f(1), f(2), \dots, f(n)\}$ . The goal is to assign the features to a finite number  $K$  of release options or to decide to postpone the feature. A release plan is characterized by a vector of decision variables  $x = (x(1), x(2), \dots, x(n))$  with

$x(i) = k$  if feature  $f(i)$  is assigned to release option  $k \in \{1, 2, \dots, K\}$ , and

$x(i) = K+1$  if the feature  $f(i)$  is postponed (e.g., not contained in one of the next  $K$  releases)

### 2.2 Stakeholders

Stakeholders are very important for performing realistic and customer-oriented release planning. We assume a set

of stakeholders  $S = \{S(1), \dots, S(q)\}$ . Each stakeholder  $S(p)$  can be assigned a relative importance  $\lambda(p)$ , based on an ordinal nine point scale. This approach is applicable to other possible scales.

### 2.3 Prioritization of Features by Stakeholders

In order to select and schedule features, there must be an agreed upon statement of priorities for features. In our model, prioritization by each stakeholder  $S(p)$  can be done with respect to different criteria. We define them here again on an ordinal nine-point scale. Possible criteria for prioritization are (but are not limited to) overall business value, urgency (time dependency), impact if feature is NOT included in a release, risk (using an inverted scale), requirements volatility (inverted scale), etc. For more information on these criteria, refer to <insert references>

### 2.4 Technological Constraints

A study of requirements repositories in the telecommunications domain by Carlshmare et al. [5] concluded that only about 20% of the features were singular or independent of each other. For this paper, we model what we consider the two most important types of technological constraints: the coupling relation  $C$ , and the (weak) precedence relation  $WP$ . Both of these relations are subsets of the product set  $F \times F$ .

**Definition 1:** Two features  $f(i)$  and  $f(j)$  are coupled (written as  $(i,j) \in C$ ) if they are required to be implemented in the same release. This dependency can be due to implementation or usage issues. In terms of the introduced decision variables, this means that

$$x(i) = x(j) \text{ for all } (i,j) \in C \subset F \times F \text{ (Coupling)} \quad (1)$$

**Definition 2:** Feature  $f(i)$  is in a (weak) precedence relation to feature  $f(j)$  (written as  $(i,j) \in WP$ ) if feature  $f(j)$  can not be delivered in a release earlier than  $f(i)$ . In terms of the introduced decision variables, this means that

$$x(i) \leq x(j) \text{ for all } (i,j) \in WP \subset F \times F \text{ (Precedence)} \quad (2)$$

### 2.5 Pre-assignment

Feature  $f(i)$  can be pre-assigned to release  $k$ , thus fixing the result of planning. In terms of the introduced decision variables, this means that

$$x(i) = k \text{ (Pre-assignment)} \quad (3)$$

### 2.6 Resource Constraints

Different resources are required for the implementation of features, and there are capacity bounds on the amount of resources available in each release cycle. We consider  $R$  types of resources involved in the implementation of features. Correspondingly, we define resource capacities  $Cap(r,k)$  for each resource type  $r = 1, \dots, R$  and all releases  $k = 1, \dots, K$ . To become a feasible plan, decision variables must satisfy

$$\sum_{x(i)=k} \text{resource}(i,r) \leq Cap(k,r) \quad (4)$$

for all releases  $k=1, \dots, K$  and all resource types  $r=1, \dots, R$ .

## 2.7 Objective Function

The objective is the maximization of a function  $F(x)$  among all release plans  $x$  satisfying the above technological and resource constraints (1) – (4).

$F(x)$  is composed of the weighted average priority vector  $WAP(j)$  defined for each feature  $f(j)$ . Therein, the weighted average priority is a function including the different possible criteria and applying operators such as +, \*, power, log, exp, Min, or Max to combine the criteria. For each release option  $k$ , parameter  $\xi(k)$  describes the relative importance of the release option and its relative impact to the objective function.

$$F(x) = \sum_{k=1 \dots K} \xi(k) [\sum_{j: x(i)=k} WAP(j)] \quad (5)$$

## 3 Handling Resource Estimation Error in the Formal Model

This paper will assume that resource overestimation errors are not nearly the problem as resource underestimation errors, and will be concerned primarily with the latter. This paper will also assume that most project development resources are time-dependent, i.e., increasing the time spent on a project increases the capacity of such resources. This would be true for most human resources on a project (assuming more personnel does not have to be added and trained, which would undermine this premise).

A third assumption made by the paper is that resource usage inefficiency and unexpected non-availability can be viewed as a form of resource estimation error. E.g, lost development time due to domain inexperience is viewed as an estimation error. Similarly, lost development time due to unexpected employee unavailability and / or attrition is also viewed as a resource estimation error, although this could technically be argued to be a different dimension of estimation error.

Given that resource underestimation errors are the most common and/or the most costly and problematic types of resource estimation errors, three approaches can be taken when resources are found to be lacking during project execution:

1. Resources can be added to the project
2. The release date of a project can be extended, which effectively increases the availability of the bulk of the resources, which are time-dependent, such as developers, testers, etc.
3. The number of features to be released can be reduced, hence reducing the amount of resources needed to develop the features.

All three approaches have their drawbacks. Approach 1, adding resources to a project, may be impossible in a given project. Even if it is not, it may be counter-productive, as Fred Brooks eloquently pointed out many years ago when he observed that “adding manpower to a late software project makes it later”[4].

Approach 2 may also be either impossible due to contractual or other obligations, or carry other severe repercussions. If there are few drawbacks to approach 2,

(deadline overruns) then it is the logical way to proceed. Unfortunately, such deadline overruns often occur even when there are severe repercussions.

Approach 3 may be the best approach when deadlines are more important than releasing all the features assigned for release on a given date. It will be the approach discussed in the remainder of this paper.

All three of these approaches can be viewed as minimizing the risk of resource estimate errors in release planning. Approach 3 can be viewed as minimizing this risk with the constraint that release dates are fixed, or nearly fixed.

Along these lines, the acceptable or even desired amount of risk may vary from project to project, and from release to release within a project. Some projects may prefer risk to be deferred to later releases to the degree possible. Gilb espoused this approach in [10]. Other projects may prefer that risk be exposed as early as possible, leading to greater risk in earlier releases. Boehm described this approach in [3]. Presumably, many of these risks, especially technical and quality risks, could be translated into, or have a strong impact on resource estimate risk.

Therefore, risk, in terms of resource estimates, shall not be handled as one of the objectives to be minimized in this paper. Rather, the discussion will center on effectively handling such risk, regardless of its magnitude.

There are two aspects of resource estimation risk. The first deals with how close an organization's resource estimates are, on average, to the actual amount of resources needed to complete a project, or perhaps a lesser unit of work such as a feature. E.g., say an organization historically completes about 85% of work within their resource estimations. Then the resource estimation risk could be said to be 15% (or -15%).

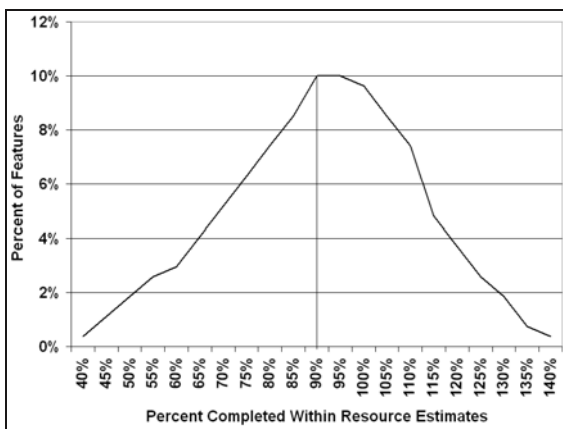


Figure 1 -- Features completed within resource estimates

The second aspect deals with how widely these estimates vary – i.e., how accurate they tend to be. Although the average historical estimate may be close to the average historical resources actually needed, the

variance may be wide, making using such estimates more risky.

Figure 1 above shows a hypothetical plot of the historical percentage of work completed for a set of features within the resource estimates for those features. Features are used instead of projects in this case because they provide a much larger sample size. Figure 1 shows that 90% of work is typically completed within resource estimates, although this can vary as much as 50% in both directions around this average.

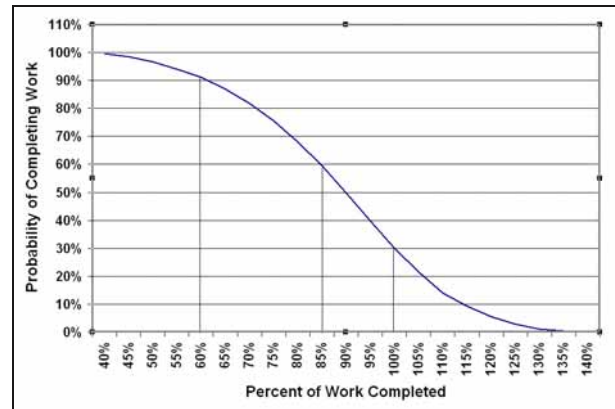


Figure 2 -- Probability of completing a percentage of work

Integration of Figure 1 from actual percent completed to 140% yields Figure 2, which shows the probability of completing a percentage of work, given the historical trends in resource estimates. For the hypothetical data set, Figure 2 shows that there is just over a 90% chance of completing 60% or more of the work within the resource estimates, just under a 60% chance of completing 85% or more of the work, and a 30% chance of completing 100% or more of the work. A somewhat similar approach is briefly discussed by Davis in [8]. However, Davis discusses the probability of releasing all features on given (flexible) dates, rather than the probability of releasing a percentage of features on a fixed date.

Given a fixed release date, a project manager can then predict, from historical resource estimation data, his or her chances of completing a certain percentage of work by the mandated release date, and plan accordingly. The approach to making these plans will be discussed in the next section.

## 4 Solution Approach

### 4.1 Execution

The first phase in executing the proposed solution would be to arrive at resource estimate probabilities, as described in the previous section. If no historical data exists, or if there are too few data points to make accurate estimates, then guessing will have to substitute. However, the proposed solution still provides significant advantages even if resource estimate probabilities are guessed at.

Once these estimates are made (or guessed at), then acceptable probability levels need to be chosen. In the

example given previously, (and to be used in the rest of this paper), 90%, 60% and 30% probability levels were chosen, which correspond to 60%+, 85%+, and 100%+ work completion levels, respectively. (There is no reason why other probability levels can not be chosen, or that the number of probability levels should be limited to 3.)

Formal release planning models can be created for any work completion estimate. Given a tool for implementing the formal model described previously (EVOLVE\*), implementing the formal models for each work completion estimate is relatively simple. Such a tool does exist, (ReleasePlanner®) [14] and will be described in the next section.

Running the formal models for the work completion estimates involves the following. First the preliminary resource estimates for all resources are reduced by a certain percentage that represents the high certainty case. In our example, the 60% completion, 90% certainty estimate is being examined, so resource estimates are reduced by 40%. This has the same effect on the model as saying that resource estimates are 40% too low. Resource estimates are reduced by 40% in both the current and subsequent releases, although the current release is our primary concern.

Release plans are then generated via automated tools (e.g., ReleasePlanner®) implementing the models, using these reduced resource estimates, and features are assigned to the defined releases by the optimization process in the models.

Next, resource capacities are expanded to encompass the medium certainty estimates. In our case, this is the 85% work completion, 60% certainty estimate solution. Again, this is done with the resource estimates for both the current and subsequent releases, although the current release is, again, our primary concern.

In addition, those features that ended up in the current release in the “60% solution” are fixed in the current release for the “85% solution” by using pre-assignment. They are denoted as belonging to the 60% solution using some appropriate notation such as “1A” or “1<sub>60</sub>”.

Release plans are then generated for the “85% solution”, after resource expansion and pre-assignment have occurred. If extra features can be added to the current release in the 85% solution, they will be added by the optimization process.

Finally, resource estimates are expanded to encompass the 30% certainty estimates. In our case, this is the 100% solution. As before, features that were added to the current release in the “85% solution” are pre-assigned to the first release, using an appropriate notation such as “1B” or “1<sub>85</sub>”.

Then release plans are generated for the 100% solution. If features can be added to the current release in the 100% solution, they will be added by the

optimization process. These features are then given an appropriate notation, such as “1C” or “1<sub>100</sub>”.

## 4.2 When Resource Estimates are too High

If resource estimates are too high, then a similar approach to what has been described can be employed. Let’s say that a project manager determines that a project is on schedule to complete with only 80% of the resources being consumed. In this case, another iteration of release planning can occur where resource estimates are expanded by 20%, in the same way that solutions were generated by contracting resource estimates in the 60, 85, and 100% solutions. New features that can be added to the current release will be added by the optimization process.

## 4.3 Monitoring by Project Management

The main intention of this approach is that, at the operational level, the features in the 60% solutions are implemented first in the current release. This ensures that they are completed and delivered even if the effort estimates fall into the (near) worst case category of being low by 40%.

Next to be completed in the current release would be features that fit into the 85% solution, assuming there is enough time to complete them. And finally, the features in the 100% solution would be completed.

This scenario allows project managers to have viable release plans even if schedules slip, but a release on a given date must be met. Managers can monitor the progress of a development organization and see which release plan seems most realistic, ahead of a release date – the high, medium, or low confidence release plans (60, 85, and 100% solutions in our example).

If the current release must contain fewer features than the 100% solution would assure, then those features that would be accepted under the 100% solution but rejected in the lower percent solutions should then normally become top priority in the next release, although this isn’t fully predictable ahead of time.

# 5 Hypothetical Case Study

## 5.1 Set-up

A hypothetical case study was run employing a benchmark release planning project staged at a large telecommunications company. The case study includes 25 candidate features to be decided upon for their release placement. Each feature is actually a kind of mini project undergoing the key steps of a development cycle (i.e., it is a very large feature). These features are documented in Table 1.

Seven resources were defined. Resource definitions are also provided in table 1, along with the consumption of the resources by the features, and the resource capacities at the “60%, 85% and 100% solution” levels, as discussed earlier (columns 4 – 10). A relatively large number of resources were employed to show that the proposed method will work with multiple resource types.

Seven hypothetical stakeholders were defined in the project, and each stakeholder voted on two criteria for each

feature – business value and urgency. These stakeholders basically served the purpose of assigning value to the features and are unimportant in this study outside of this role. Therefore, they are not documented in this paper.

There were two dependencies between the features – one coupling dependency – between features 9 and 10, and one precedence dependency, between features 14 and 15. Originally, no features were pre-assigned to releases. Pre-assignments to the current release occurred when 85% and 100% solutions were determined, as described previously. Pre-assignments are shown in Table 1 as dotted cells in the release placement columns.

## 5.2 Results

Release planning solutions were generated at the 60%, 85%, and 100% resource capacity levels by the ReleasePlanner® support tool. The placement of features in releases can be seen in the three right-most (release placement) columns in Table 1.

At the 60% level, 7 features were placed in release 1 (the current release) by the tool. These features are labeled in the table as being in release “1A” to indicate that they should be implemented first, to allow planning for the cases where resource capacities might be expanded to the 85% or 100% levels.

All of the 1A features were then pre-assigned to release 1, and resource capacities were expanded to 85%, and the ReleasePlanner® tool was executed again. At this level, two more features were added to the current release by the tool, (features 2 and 22) and are labeled “1B” in the table. The 1B features were then also pre-assigned to release 1, resource capacities were expanded to 100%, and the tool was run again. At this level, two more features were placed in release 1 (features 14 and 24), and are labeled “1C” in the table.

Feature placement in release 2 is not that critical, because it is assumed that when the current release (release 1) is completed, then the next release (release 2) will be subject to re-planning anyway, with multiple new factors in mind. However, placement of features in release 2 can provide a general roadmap for future development.

In general, it would be expected that any “1B” features in an 85% or 100% solution would end up in release 2 of a 60% solution. Similarly, any “1C” features in a 100% solution would end up in release 2 of a 60% or 85% solution. In the sample project, this happened in all cases but one. In that one case, a “1C” feature (feature 24) in a 100% solution ended up being postponed in the 85% solution, yet placed in release 2 of the 60% solution. The reasons for this are not

immediately apparent, but when resource capacities are expanded in the 85% solution, the optimization algorithm saw a better opportunity for organizing the features by postponing feature 24. The multi-dimensional nature of the resource capacities makes analyzing the exact reason difficult.

## 5.3 Discussion

The one major caveat of this approach lies in the implicit assumption that the features are implemented in a serial fashion. This is often not the case, and features are often implemented in parallel, with different developers working on different features simultaneously.

If a release plan has many high confidence solution features to be implemented, but only a few medium and low confidence solution features, there may be problems fully utilizing a staff to implement the small number of medium and low confidence solution features.

Another caveat lies in the fact that some formal models (including EVOLVE\*) can handle the modeling of multiple resources simultaneously, and different resource estimates may be off by different amounts. However, EVOLVE\* can still create formal models using the approach described in this paper.

Handling both of these situations fall into the domain of operational rather than strategic planning, and deal with the allocation of resources at the feature level, which is outside the scope of this paper.

## 6 Summary and Conclusions

Formal release planning has the advantage of offering optimized solutions for projects of almost any scale, in which features are assigned to releases such that most important technical, resource and budget constraints are met. In addition, formal release planning models can handle complexity far beyond what human beings are capable of handling.

However, formal release planning produces solutions inferring accuracy that may be beyond what the inputs to the model warrant, especially when fixed release dates and/or fixed resources are mandated.

A solution to this problem was presented in this paper, in which sets of release plans are generated for low, medium, and high certainty feature completion given fixed resources and fixed release dates. This allows project managers to have working sets of release plans, even if significant schedule slippage occurs.

## Acknowledgment

The authors would like to thank the Natural Sciences and Engineering Research Council (NSERC) (Discovery Grant 250343-07) for the financial support of this research.

Table 1 -- Features, varying resource capacities and layered release strategies.

Features			Resource Consumption (In days, except for Capital Req.)							Release Placement		
#	ID	Name	BTS SW Dev.	BTS HW Dev.	BSC/BS M SW Dev.	MTX SW Dev.	Test-ers	Doc Writ-ers	Capital Req (\$k)	60%	85%	100%
1	BTS-HW01	Cost Reduction of Transceiver	150	200	120	0	200	60	1000	1A	1A	1A
2	BTS-HW02	16 sector, 12 carrier BTS for China	400	300	150	150	200	150	1000	2	1B	1B
3	BTS-HW03	Expand Memory on BTS Controller	75	120	10	0	75	20	200	1A	1A	1A
4	BTS-HW04	Next Generation BTS 'In a Shoebox'	450	350	375	125	500	200	150	pp	pp	2
5	BTS-HW05	Pole Mount Packaging	400	180	300	50	400	150	500	pp	2	pp
6	BTS-HW06	FCC Out-of-Band Emissions Reg. Change	400	120	100	0	200	10	200	1A	1A	1A
7	BTS-HW07	India BTS variant	575	420	400	200	250	200	750	pp	pp	pp
8	BTS-SW01	Software Quality Initiative	450	0	100	50	400	5	0	pp	2	2
9	SYS-SW01	USEast Inc. Feature 1	100	0	400	100	40	100	0	1A	1A	1A
10	SYS-SW02	USEast Inc. Feature 2	200	0	400	150	50	50	25	1A	1A	1A
11	SYS-SW03	USEast Inc. Feature 3	400	0	100	100	40	20	100	pp	2	2
12	SYS-SW04	USEast Inc. Feature 4	150	0	400	125	400	150	1000	pp	pp	2
13	SYS-SW05	USEast Inc. Feature 5	75	180	225	225	300	60	750	pp	pp	pp
14	SYS-SW06	China Feature 1	50	0	250	140	200	60	500	2	2	1C
15	SYS-SW07	China Feature 2	60	10	120	120	190	40	200	2	2	2
16	SYS-SW08	China Feature 3	75	75	300	120	450	50	500	pp	pp	pp
17	SYS-SW09	China Feature 4	0	0	100	150	100	50	0	1A	1A	1A
18	SYS-SW10	China Feature 5	250	100	400	400	400	50	300	pp	pp	pp
19	SYS-SW11	India Mkt Entry Feature 1	200	100	250	250	250	100	500	pp	pp	pp
20	SYS-SW12	India Mkt Entry Feature 2	0	0	300	250	250	100	300	pp	pp	pp
21	SYS-SW13	India Mkt Entry Feature 3	100	100	150	100	300	25	1200	1A	1A	1A
22	SYS-SW14	Common Feature 01	100	0	250	100	200	0	50	2	1B	1B
23	SYS-SW15	Common Feature 02	0	0	100	250	150	50	0	pp	2	2
24	SYS-SW16	Common Feature 03	200	0	150	0	100	20	0	2	pp	1C
25	SYS-SW17	Common Feature 04	100	0	300	200	200	30	50	pp	pp	pp
Total Possible Consumption			4960	2255	5750	3355	5845	1750	9275			
<b>Resource Capacities</b>										<b>pp = Postponed</b>		
100% solution release 1			1800	1100	2160	960	1680	600	4800			
100% solution release 2			1600	960	1680	960	1680	480	4800			
85% solution release 1			1530	935	1836	816	1428	510	4080			
85% solution release 2			1360	816	1428	816	1428	408	4080			
60% solution release 1			1080	660	1296	576	1008	360	2880			
60% solution release 2			960	576	1008	576	1008	288	2880			
<b>Resource Utilization</b>												
100% solution release 1			98.6%	76.4%	96.3%	97.2%	99.1%	90.8%	87.0%			
100% solution release 2			94.4%	37.5%	71.1%	80.2%	100.0%	96.9%	30.2%			
85% solution release 1			99.7%	89.8%	91.5%	91.9%	95.6%	91.2%	90.1%			
85% solution release 2			100.0%	23.3%	67.9%	87.0%	96.6%	79.7%	31.9%			
60% solution release 1			94.9%	81.8%	98.8%	86.8%	95.7%	87.5%	91.1%			
60% solution release 2			84.4%	53.8%	81.3%	88.5%	88.3%	93.8%	60.8%			

**References**

[1] Bagnall, A. J., Rayward-Smith, V. J., and Whittle, I. M., "The Next Release Problem", Information and Software Technology, Vol. 43 (2001), 883-890.

[2] Beck, K., "Extreme Programming Explained", Addison Wesley, 2001.

[3] Boehm, B., "A Spiral Model of Software Development and Enhancement," Proceedings of the International Workshop Software Process and Software Environments, ACM Press, 1985.

[4] Brooks, F. "The Mythical Man Month"

[5] Carlshamre, P., "Release Planning in Market-driven Software Product Development - Provoking an Understanding", Journal of Requirements Engineering, Vol. 7, No. 3, 2002, 139-151.

[6] Cohn, M., "User Stories Applied: For Agile Software Development" 2004, Prentice-Hall.

[7] Cohn, M., "Agile Estimating and Planning", 2005, Prentice-Hall.

[8] Davis, A., The Art of Requirements Triage. IEEE Computer 36 (3), pp. 42- 49. 2003.

[9] Du, G., McElroy, J., and Ruhe, G., "Ad hoc versus Systematic Planning of Software Releases - A Three-Stage Experiment", Proceedings of 7th International Conference on Product Focused Software Process Improvement (PROFES06), Amsterdam 2006.

[10] Gilb, T., "Principles of Software Engineering Management", Addison Wesley Longman, 1989.

[11] Jung, H-W., "Optimizing Value and Cost in Requirements Analysis", IEEE Software, Vol. 15 (1998), No 4, 74-78.

[12] McBreen, P. 2003. Questioning Extreme Programming. Boston: Addison-Wesley; p. 100.

[13] A. Ngo-The, G. Ruhe, "Optimized Resource Allocation for Incremental Software Development", IEEE TSE, January/February 2009 (vol. 35 no. 1) pp. 109-123.

[14] [www.releaseplanner.com](http://www.releaseplanner.com)

[15] Ruhe, G., and Ngo-The, A., "Hybrid Intelligence in Software Release Planning", International Journal of Hybrid Intelligent Systems, Vol. 1, No. 2, 2004, 99-110.

[16] Ruhe, G., and Saliu, O., "The Art and Science of Software Release Planning", IEEE Software, Vol. 22 (2005), No. 6, 47-53.

[17] van den Akker, M., Brinkkemper, S., Diepen, G. and Versendaal, J., "Software product release planning through optimization and what-if analysis", Information and Software Technology 50, pp. 101-111, 2008.

[18] van-Gurp, J., Bosch, J, and Svahnberg, M, "Managing Variability in Software Product Lines", Proceedings of LAC'2000, Amsterdam.

# PP-HAS: A Task Priority Based Preemptive Human Resource Scheduling Method

Lizi Xie<sup>1,3</sup>, Qing Wang<sup>1</sup>, Junchao Xiao<sup>1</sup>, Yongji Wang<sup>1,2</sup>, Ye Yang<sup>1</sup>

<sup>1</sup> Laboratory for Internet Software Technologies, Institute of Software,  
The Chinese Academy of Sciences, China

<sup>2</sup> Key Laboratory for Computer Science,  
The Chinese Academy of Sciences, China

<sup>3</sup> Graduate University of Chinese Academy of Sciences, China  
{xielizi, wq ,xiaojunchao,ywang,ye }@itechs.iscas.ac.cn

## Abstract

*To schedule human resource effectively is an important research topic in software project management field. Optimized scheduling for limited resource is a firm assurance for software project success. The most important problem is how to ensure that the more valuable task be satisfied with resource when resource is not enough for all the tasks. Traditional human resource scheduling mainly depends on the project manager's experiences and instincts. Human resource scheduling is much more difficult for those software companies which have many concurrent software projects. Resource conflicts often take place and make great trouble to the management work. There is a lack of scheduling method to support software project manager's work under limited human resource. In this work, we propose a preemptive human resource scheduling method base on task priority and Process-Agents' negotiation. The capability and work time of the assigned human resource can be guaranteed. The value-based task priority model introduced in this paper is integrated with Process-Agents to supply decision support for project managers who are struggling against resource conflicts in software projects. The method can help software companies gain high human resource utilization rate and improve their software project management capability.*

## 1. Introduction

Human factors are much more important for software process compared with traditional industry process, and human resource is the core resource in software development<sup>[1]</sup>. The purpose of researching resource scheduling in software projects is to help software companies make better use of their limited human resource. Schedule, quality and cost of a software project can be controlled only when the resource are scheduled clearly and effectively.

Traditional human resource scheduling mainly depends on the project manager's experiences and

instincts. It works well when the project and the organization are particularly small. Actually the environment is often very complicated. To schedule human resource effectively is much more difficult in Multi-Projects development. There is a lack of scheduling method for limited human resource to support software project management work.

Boehm has proposed VBSE (*Value based Software Engineering*)<sup>[2]</sup>. He claims that the factors in software engineering are not value-neutral. 20% of the features provide 80% of the value. Pareto Principle<sup>[3]</sup> should be considered in software engineering. We have to admit that the personal capability is different from person to person, and the task's value for the organization is also different from task to task. In order to make better use of limited human resource, two problems should be solved: How to describe and evaluate the value of tasks for the organization and how to make human resource scheduling much more effective and flexible.

We propose preemptive human resource scheduling through Process-Agents' negotiation, and using value based task priority to solve resource conflicts. First, value based task priority model can decompose a project's value to its tasks through the three dimensions (*Schedule, Quality, Cost*). The value of the task is consistent with the project status. Second, a preemptive human resource scheduling method is proposed. The resource can be allocated or reallocated much more smoothly according to the inconstant project environment.

The rest of this paper is organized as follows, section 2 introduces some related works, section 3 will discuss the value based task priority model and describes the resource scheduling method with Process-Agents, an example of the method is shown in section 4. Conclusion and future work are presented in section 5.

## 2. Related Work

Human, technology and process are the three core factors in software production. How to describe,



manage and schedule human resource effectively is the key of software project management. We have proposed OEC-SPM (*Organization Entity Capability based Software Process Modeling*) [4]. The organization resource and knowledge asset are organized by a set of Process-Agents. Process-Agent can self-government and cooperate with each other. The work presented in this paper is based on OEC-SPM, and focuses on optimized human resource scheduling.

QONE [5] has been widely used by software companies in china. We have developed a tool [6] to create Process-Agents automatically from historical project data in QONE. The work in this paper enhances the relationship between Process-Agents and human resource.

Scheduling is to allocate resource and time for a shared goal. There are many scheduling arithmetic in operations research fields [7]. But these standard theoretical methods mainly focus on traditional industry process where the output of the equipments often plays the key role. The resource are defined and scheduled without considering the complexity and particularity of human. Our former research [8] aims to optimize human resource allocation for single software project, but it does not take multi-projects environment for consideration.

### 3. Task Priority based Preemptive Human Resource Scheduling Method

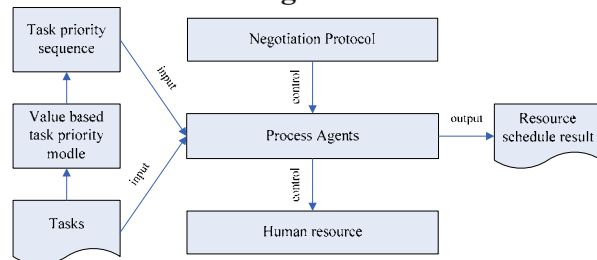


Figure 1. The main framework of PP-HAS

Figure1 shows the main framework of our resource scheduling method. The input of the method is a task set and a task priority sequence which is produced by the task priority model. Process-Agents will schedule human resource following the given negotiation protocol. When resource conflicts occur, decisions will be made by Process-Agent base on task priority sequence. The output of the method is the optimized resource schedule result which can do decision support for software project managers.

### 3.1 Value based Task Priority Model

#### 3.1.1 Model Definitions

Project value indicates the importance of the software project to the organization. For example, we can simply using this formula to estimate the value of a project:

$$P.v = (R - C) \times PS$$

(R: Expected return from market. C: Estimated cost of the project. PS: Probability of the project success).

Table 1. The explanations of the parameters in task priority model

Parameter	Explanation
$P$	Project
$P.v$	The value of the project
$FS$	The set of factors which can affect $P.v$ $FS = \{P.S, P.Q, P.C\}$
$P.S$	The schedule factor which can affect $P.v$
$P.Q$	The quality factor which can affect $P.v$
$P.C$	The cost factor which can affect $P.v$
$P.I_S$	The importance rank of $P.S$ to $P$
$P.I_Q$	The importance rank of $P.Q$ to $P$
$P.I_C$	The importance rank of $P.C$ to $P$
$P.W_S$	The control power of $P$ on schedule
$P.W_Q$	The control power of $P$ on quality
$P.W_C$	The control power of $P$ on cost
$T$	Task
$T.s$	The schedule factor of task $T$
$T.I_S$	The degree of the impact for $T.s$ to $P.S$
$T.q$	The quality factor of task $T$
$T.I_Q$	The degree of the impact for $T.q$ to $P.Q$
$T.c$	The cost factor of task $T$
$T.I_C$	The degree of the impact for $T.c$ to $P.C$

Software project management relates to schedule management, quality management and cost management. To get a balance among the three dimensions in appropriate scope is the core to ensure software project success. The importance of **S**, **Q** and **C** \* varies from project to project because of different business goals and organization environments. In order to manage projects effectively, we have to order the importance of the three dimensions for a project. In our work, we try to analyze and decompose a project's value through the three dimensions. The symbols used in the model are listed and explained in table1.

\* In the rest parts of this paper, S, Q and C are short for Schedule, Quality and Cost.

**Task Value Definition:**  $T_x.v$  is the value of task  $T_x$ .

$$T_x.v = \left( \frac{T_x.s \times T_x.ts}{\sum (T_n.s \times T_n.ts)} \times P.sf + \frac{T_x.q \times T_x.tq}{\sum (T_n.q \times T_n.tq)} \times P.qf + \frac{T_x.c \times T_x.tc}{\sum (T_n.c \times T_n.tc)} \times P.cf \right) \times P.v$$

$P.sf$  is the **value proportion** for P.S to P.

$$P.sf = \frac{P.I_s \times P.W_s}{P.I_s \times P.W_s + P.I_q \times P.W_q + P.I_c \times P.W_c}$$

$P.qf$  is the **value proportion** for P.Q to P.

$$P.qf = \frac{P.I_q \times P.W_q}{P.I_s \times P.W_s + P.I_q \times P.W_q + P.I_c \times P.W_c}$$

$P.cf$  is the **value proportion** for P.C to P.

$$P.cf = \frac{P.I_c \times P.W_c}{P.I_s \times P.W_s + P.I_q \times P.W_q + P.I_c \times P.W_c}$$

Through the definition of  $P.sf$ ,  $P.qf$  and  $P.cf$ , we can decompose project value through S/Q/C. As we all know, a project consists of several tasks. Task here means the low level work package in WBS [9], its cost, workload, duration, work-product and resource requirement are finely estimated. We will evaluate the impact of a task for the project from the three aspects (S/Q/C), and decompose the project's value into task's value by choosing reasonable weights (see Task Value Definition). It can be easily proved that  $\sum T_x.v = P.v$

We can compare the priorities of tasks by their values. Tasks with higher value will gain high priority; tasks with lower value will gain lower priority.

### 3.1.2 Quantify the Parameters

We can embody the model by implementing the five rules below. The method we finally get can help project manager to estimate the value of different tasks. The approaches of quantifying the parameters can be customized according to the organization situation.

**Rule<sub>1</sub>. Quantify the importance of S, Q and C for a project.**

$$(1 \leq P.I_s \leq 10) \wedge (1 \leq P.I_q \leq 10) \wedge (1 \leq P.I_c \leq 10) \wedge (P.I_s + P.I_q + P.I_c = 10)$$

The value of the three parameters in rule<sub>1</sub> can be decided by the managers using Delphi Method.

**Rule<sub>2</sub>. Quantify the control power on S, Q and C for a project.**

IF the allowed deviation of S/Q/C is below 3%,  $P.W_s / P.W_q / P.W_c = 4$ ; IF the allowed deviation of S/Q/C is in the range (3%,10%],  $P.W_s / P.W_q / P.W_c = 3$ ; IF the allowed deviation of S/Q/C is in the range (10%,30%],  $P.W_s / P.W_q / P.W_c = 2$ ; IF the allowed deviation of S/Q/C is higher than 30%,  $P.W_s / P.W_q / P.W_c = 1$ .

For example, if the allowed deviation of S is 2%, the allowed deviation of Q is 15%, the allowed deviation

of C is 25%, then  $P.W_s = 4, P.W_q = 3, P.W_c = 2$ . The deviation range and the value of weight can be modified by the user to reflect their specific situation.

**Rule<sub>3</sub>. Quantify task schedule factors**

$T.ts$  is the planed duration of the task, the unit should be "day".

If  $T$  is on the critical path of the project,  $T.ts = 2$ , else,  $T.ts = 1$ .

The tasks on the critical path are much more important for project schedule.

**Rule<sub>4</sub>. Quantify task quality factors**

$T.tq$  is the number of estimated defects which will be injected by task  $T$ .

If  $T$  belongs to the "Requirement" phase of the project life,  $T.tq = 4$ . If  $T$  belongs to the "Design" phase of the project life,  $T.tq = 3$ . If  $T$  belongs to the "Coding" phase of the project life,  $T.tq = 2$ . If  $T$  belongs to the "Test" phase of the project life,  $T.tq = 1$ .

We believe that the earlier the defect was injected, the more important it is. The phases of the project life can be extended if needed.

**Rule<sub>5</sub>. Quantify task cost factors**

$T.tc$  is the plan cost of the task  $T$ . The unit could be \$.

For all the tasks,  $T.tc = 1$ .

### 3.1.3 An Example of Using the Model

The software company has many customized versions of a product to maintain. Two customers have raised some new function requirements, so there are two projects  $p_1$  and  $p_2$ . Because of the time and human resource limitation, there will be resource conflicts between the tasks of the two projects. The project manager has to decide which task is more valuable so that it should be assigned enough human resource with high priority. The manager has estimated the value of these two projects:  $p_{1.v} = 360, p_{2.v} = 460$ . Now, we will calculate the task priority sequence in the two projects using the task priority model.

At first, we will decompose the value of each project on the three dimensions (S/Q/C) using rule1 and rule2. The result of  $p_1$  is shown in table2, and the result of  $p_2$  is shown in table 3.

The work of the two projects has been broke down by the project manager. The task attributes have been finely estimated, such as the plan duration, plan cost, estimated defect injected and so on. The task attributes in both projects are shown in Table 4. We can get the parameters for each task by using rule3-5. (Table 5).

**Table 2. The parameters of  $P_1$  (F(Factor) IR(Importance Rank) VP(Value Proportion))**

F	IR	Allowed deviation %				VP
		≤3	3-10	10-30	>30	
S	5	√	-	-	-	P.sf=0.61
Q	3	-	√	-	-	P.qf=0.27
C	2	-	-	√	-	P.cf=0.12
Sum	10	-	-	-	-	1

**Table 3. The parameters of  $P_2$  (F(Factor) IR(Importance Rank) VP(Value Proportion))**

F	IR	Allowed deviation %				VP
		≤3	3-10	10-30	>30	
S	3	-	√	-	-	P.sf=0.32
Q	5	-	√	-	-	P.qf=0.54
C	2	-	-	√	-	P.cf=0.14
Sum	10	-	-	-	-	1

**Table 4. Tasks in the two projects. (C (Cost), D (Plan Duration), CP (Critical Path), PH (Phrase), DI (Estimated defects injected))**

Project	Task	C	D	CP	PH	DI
$P_1$	T <sub>1</sub>	1600	5	Y	Design	10
	T <sub>2</sub>	1280	4	Y	Test	3
	T <sub>3</sub>	1600	5	N	Code	5
	T <sub>4</sub>	1920	6	Y	Code	6
$P_2$	T <sub>5</sub>	2240	7	Y	Design	30
	T <sub>6</sub>	2560	8	Y	Test	10
	T <sub>7</sub>	3200	10	Y	Code	15

**Table 5. The derived parameters of tasks**

T	$T_s$	$T_{ts}$	$T_q$	$T_{tq}$	$T_c$	$T_{tc}$
T <sub>1</sub>	5	2	10	3	1600	1
T <sub>2</sub>	4	2	3	1	1280	1
T <sub>3</sub>	5	1	5	2	1600	1
T <sub>4</sub>	6	2	6	2	1920	1
T <sub>5</sub>	7	2	30	3	2240	1
T <sub>6</sub>	8	2	10	1	2560	1
T <sub>7</sub>	10	2	15	2	3200	1

Now, we can get the value of each task by using *Task Value Definition* in section 3.1.1

$$T_{1,v} = 126.828, T_{2,v} = 63.936, T_{3,v} = 59.94, T_{4,v} = 109.332, T_{5,v} = 230.414, T_{6,v} = 87.308, T_{7,v} = 142.278$$

Based on the task values, we can get the task priority sequence:  $T_5 > T_7 > T_1 > T_4 > T_6 > T_2 > T_3$ . This task priority sequence can do much help for us when scheduling human resource.

### 3.2 The Process-Agents based Preemptive Resource Scheduling Method

#### 3.2.1 Concepts of Process-Agent

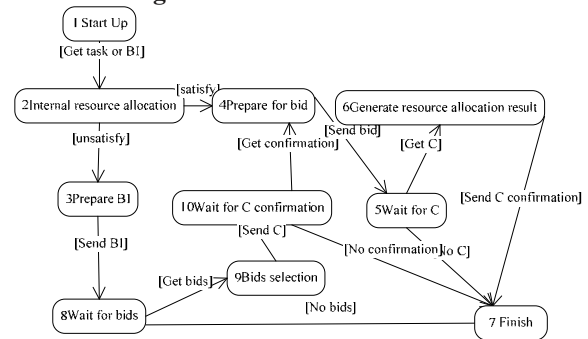
PA (*Process-Agent*) is first introduced in OEC-SPM<sup>[6][7]</sup>. PA is the abstract description of a set of organizational

human resources with similar capability. The resource scheduling method in this paper is based on these PAs.

PA consists of three kinds of knowledge and some engines. The engines control the self-government actions and interactions of the Process- Agents. Description knowledge (DK) indicates the resource owned by the PA, what can PA do, and the capability of the PA. Process knowledge (PK) indicates the detailed processes for PA to achieve special goals. Experience Lib (EL) holds the history data of executing software processes.

We extended DK of PA. Human resource maintains the unit cost and work calendar. Different human resources have different unit costs and work calendars.

#### 3.2.2 Resource Scheduling Process base on Negotiation



**Figure 2. Resource scheduling process of PA (BI (Bid Invitation), C (Contract))**

Because human resource is organized by a group of PAs, the resource scheduling process exists not only inside a PA, but also the cooperation among PAs. PA will negotiate and cooperate with other PAs to satisfy a task when its own resource is not enough. Figure 2 shows the process of a PA when conducting resource scheduling and negotiating with others.

The information space for PAs to communicate with each other is called *shared information space (SIS)*. The sub resource requirement formed by PA is called *bid invitation (BI)*.

The resource scheduling and negotiation process of PA are explained as follows:

**Step 1. Start Up:** After being started, PA will perceive the SIS to find new tasks or bid invitations. PA will judge whether it has the capability to do the task or the bid invitation according to its knowledge, if so, PA will turn to Step 2, else, it will continue perceiving the SIS.

**Step 2. Internal Resource Scheduling:** PA analyzes the resources needed by the task or bid invitation, searches the optimized human resource combination in its human resource set. When necessary, resource

assigned to other tasks of lower priority will be robbed to satisfy high priority task. If PA can find suitable resource combination, it turns to Step 4; else, it turns to Step 3.

When searching suitable resource combination, if the available free human resource is not enough, Task priority model will be used by PA to do preemptive human resource scheduling. First, when searching suitable human resource combination for the given task, free human resources and human resources occupied by lower priority tasks compared with the target task are both considered. Second, if there is more than one tasks whose resource can be robbed, PA will choose the task of lowest priority.

**Step 3. Bid Invitation Preparation:** PA forms a bid invitation based on the lack of resource. After sending it to the shared information space, PA turns to Step 8.

**Step 4. Bid Preparation:** PA forms a bid based on the selected human resource combination, sends it to the share information space, and turns to Step 5.

**Step 5. Waiting for Contract:** PA searches SIS to find corresponding contract. If the contract is found, PA turns to Step 6; else, PA turns to Step 7.

**Step 6. Generating Resource Plan:** PA generates resource plan according to the contract, updates the work calendars of the human resource, sends contract confirmation to SIS, and turns to Step 7.

**Step 7. Finish:** PA finishes the resource scheduling process, waits for the next startup.

**Step 8. Waiting for Bid:** PA searches the SIS, if expected bids are found, it turns to Step 9, and else, it turns to Step 7.

**Step 9. Bid Selection:** PA selects the best bid, forms a contract based on the chosen bid, sends the contract to SIS, and turns to Step 10.

**Step 10. Waiting for Contract Confirmation:** PA searches the SIS, if expected contract confirmation exists, it turns to Step 4, and else, it turns to Step 7.

We will give an example to illustrate the method in the following section.

## 4 An Example of the Method

### 4.1 Case Introduction

We will take the two projects in section 3.1.3 to illustrate the process of task priority based preemptive human resource scheduling method. From section 3.1.3 we know that the task priority sequence is: **T5>T7>T1>T4>T6>T2>T3**. There are nine PAs in

our system listed in Table 6. (The description of capability of PA is simplified compared to OEC-SPM. More detail please refer to <sup>[4]</sup>)

**Table 6. Process-Agents in the system**

Name	Capability	Rank	Resources
PA <sub>1</sub>	Design	Senior	HR <sub>1</sub>
PA <sub>2</sub>	Design	Medium	HR <sub>2</sub>
PA <sub>3</sub>	Design	Junior	HR <sub>3</sub>
PA <sub>4</sub>	Code	Senior	HR <sub>4</sub>
PA <sub>5</sub>	Code	Medium	HR <sub>5</sub> HR <sub>6</sub>
PA <sub>6</sub>	Code	Junior	HR <sub>7</sub> HR <sub>8</sub>
PA <sub>7</sub>	Test	Senior	HR <sub>9</sub>
PA <sub>8</sub>	Test	Medium	HR <sub>10</sub>
PA <sub>9</sub>	Test	Junior	HR <sub>11</sub> HR <sub>12</sub>

Supposing that project1 has already finished resource allocation (table7). Now the emergent project2 has to be set up(table8). There will be resource conflicts between the two projects. The resource should be rescheduled to satisfy the two projects.

**Table 7. Tasks in project 1 (T (Task), C (Capability), NH (Number of human required), PSD (Plan Start Date), PFD (Plan Finish Date))**

T	C	NH	PSD	PFD	HRs
T <sub>1</sub>	D	1	2009-5-1	2009-5-5	HR <sub>1</sub>
T <sub>3</sub>	C	2	2009-5-6	2009-5-12	HR <sub>6</sub> HR <sub>7</sub>
T <sub>4</sub>	C	2	2009-5-6	2009-5-13	HR <sub>4</sub> HR <sub>5</sub>
T <sub>2</sub>	T	2	2009-5-14	2009-5-18	HR <sub>9</sub> HR <sub>10</sub>

**Table 8. Tasks in project 2**

T	C	NH	PSD	PFD
T <sub>5</sub>	D	1	2009-4-29	2009-5-5
T <sub>6</sub>	T	2	2009-5-16	2009-5-23
T <sub>7</sub>	C	2	2009-5-6	2009-5-15

We will schedule the resource for the two projects using our method under the situation described above.

### 4.2 The Process and Result of Resource Scheduling

There is a special PA (PA-System) who is in charge of the new tasks. All the tasks will be scheduled according their logic sequence. We will take the resource scheduling process for task T<sub>7</sub> for example to illustrate our method:

**Step1:** PA-System puts T<sub>7</sub> into share information space. PA<sub>4</sub>, PA<sub>5</sub> and PA<sub>6</sub> will try to satisfy T<sub>7</sub> based on T<sub>7</sub>'s capability requirement.

**Step2:** PA<sub>4</sub> finds that human resource HR<sub>4</sub> is occupied by T<sub>4</sub> in the time period (2009-5-6, 2009-5-15). Because T<sub>7</sub>' priority is higher than T<sub>4</sub>'s, HR<sub>4</sub> can be robbed by T<sub>7</sub>. But PA<sub>4</sub> can only offer one resource for T<sub>7</sub> while two persons are needed. PA<sub>4</sub> will form a bid invitation based on the resource lack. The bid invitation indicates that it needs one person with designing capability during (2009-5-6, 2009-5-15),

and it is derived from  $T_7$ .  $PA_4$  puts the bid invitation into SIS.

**Step3:**  $PA_5$  and  $PA_6$  can detect the bid invitation. They will try to satisfy it.  $PA_5$  finds that its two resources are both occupied. Based on task priority ( $T_7 > T_4 > T_3$ ),  $PA_5$  decides to rob  $T_3$ 's resource.  $PA_5$  forms the bid {2009-5-6, 2009-5-15, HR6}.

**Step4:**  $PA_6$  forms the bid {2009-5-6, 2009-5-15, HR8}.

**Step5:**  $PA_4$  will choose  $PA_5$ 's bid, because  $PA_5$  has higher capability than  $PA_6$ .

**Step6:** After contract and contract confirmation,  $PA_5$  updates  $HR_6$ 's work calendar,  $PA_4$  integrates its bid with  $PA_5$ 's. The new bid is {2009-5-6, 2009-5-15, HR4, HR6}.

**Step7:**  $PA_5$  and  $PA_6$  will form its bids too, but only  $PA_4$ 's bid will be chosen by PA-System based on their capability level.

The four times of resource scramble in the whole scheduling process are summarized in Table 9. The final resource scheduling result is shown in table 10.

**Table 9. The four times of resource scramble**

ID	Resource scramble detail
1	$T_5$ Robs $T_1$ of $HR_1$
2	$T_7$ Robs $T_4$ of $HR_4$
3	$T_7$ Robs $T_3$ of $HR_6$
4	$T_6$ Robs $T_2$ of $HR_9$ and $HR_{10}$

**Table 10. The final result of resource scheduling**

Task	Process-Agent	Resources
$T_1$	$PA_2$	$HR_2$
$T_2$	$PA_9$	$HR_{11}$ $HR_{12}$
$T_3$	$PA_6$	$HR_7$
$T_4$	$PA_5$ $PA_6$	$HR_5$ $HR_8$
$T_5$	$PA_1$	$HR_1$
$T_6$	$PA_7$ $PA_8$	$HR_9$ $HR_{10}$
$T_7$	$PA_4$ $PA_5$	$HR_4$ $HR_6$

From the resource scheduling result in table 10, the advantages of our method can be summarized:

1. Based on Process-Agent, the human resource capability is finely considered when scheduling resource. The most skillful human resources are assigned to the most important tasks. ( $T_5$  got the most skillful person). And the assigned human resource will have enough time for task execution.
2. When resource is not enough, the task with higher value will be satisfied first. The total income with limited resource can be maximized. (The total income is 760.06 ( $T_1.v + T_2.v + T_4.v + T_5.v + T_6.v + T_7.v$ ) better than the income under un-preemptive schedule method: 677.722 ( $T_1.v + T_2.v + T_3.v + T_4.v + T_5.v + T_6.v$ ))

There might be a plan change requirement for  $T_3$ . The project manager will focus on dealing with the task of lowest value.

## 5 Conclusion and Future Work

Preemptive human resource scheduling method can help project managers to optimize human resource utilization in the complicated Multi-Projects environment. Using Process-Agent can support effective resource scheduling under human capability. PP-HAS can help software companies optimize resource management and solve resource conflicts. We believe that making better use of human resources will greatly improve software companies' enterprise competitive power. Future work will focus on developing tools and refining the task priority model.

**Acknowledgments:** Supported by the National Natural Science Foundation of China under grant Nos. 90718042, the Hi-Tech Research and Development Program (863 Program) of China under grant No.2007AA010303, 2007AA01Z186, as well as the National Basic Research Program (973 program) under grant No. 2007CB310802.

## 7 References

- [1] Kathy schwalbe, Information technology project management, 2<sup>nd</sup> edition
- [2] Barry Boehm , A Value-Based Software Process Framework, SPW/ProSim2006,Shaing,China
- [3] Richard Koch. The 80/20 Principle: The Secret to Success by Achieving More with Less. Doubleday Business. 1998
- [4] Qing Wang, Junchao Xiao, Mingshu Li, M. Wasif Nisar, Rong Yuan, and Lei Zhang. A PA Construction Method for Software Process Modeling in SoftPM. SPW/ProSim 2006, LNCS 3966
- [5] Q. Wang, M. Li: Software Process Management: Practices in China. SPW 2005, LNCS 3840, pp. 317–331
- [6] Lei Zhang, Qing Wang, Junchao Xiao, Li Ruan, Lizi Xie, Mingshu Li, A Tool to Create PAs for OEC-SPM from Historical Project Data :ICSP2007, LNCS 4470, pp.84-95
- [7] Brucker P. Scheduling Algorithms. Springer Verlag. 2001.
- [8] Lizi Xie. A Project Scheduling Method Based on Human Resource Availability. SEKE2008.
- [9] IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans

# A Real Execution of a Software Process Improvement: An Opportunity to Execute a Combination of Approaches

<sup>1</sup>Adriano Bessa Albuquerque, <sup>2</sup>Ana Regina Rocha

<sup>1</sup>University of Fortaleza (UNIFOR) - Washington Soares Avenue, 1321 - Bl J Sl 30 - 60.811-341 – Fortaleza, Brazil

<sup>2</sup>COPPE/UFRJ - Federal University of Rio de Janeiro, POBOX 68511 – ZIP21945-970 – Rio de Janeiro, Brazil

## Abstract

*Software development organizations must improve their products' quality, increase their productivity, reduce the projects' costs and increase the projects' predictability aiming to maintain their competitiveness in the market. So, it is essential to invest on software process and support approaches to continually improve the processes on this volatile market. This paper presents the execution of a process to evaluate and improve the organizational process assets in a software organization, where we can experiment a combination of known approaches of different knowledge areas. At the end, we analyzed the results and suggested some improvements to the process.*

## 1. Introduction

Nowadays, the world's software industry increases because the software becomes part of many products and activities. In the United States, between 1995 and 1999, the tax of investments in software was four times higher than in the period of 1980-85 [1].

However, the software organizations need improve the quality of their products, increase the productivity, reduce the costs and increase the predictability of the projects to continue in this promising market, where the changes of the clients needs are constants and the increase of the competitiveness.

In face of this context and of the knowledge that the quality of the software products is influenced by the quality of the software processes used to develop them [2], the industry of software and the academy are investing more and more in researches related to

software process. Besides, as the market is volatile and its level of exigency increases day by day, the processes should stay all the time in a state of continuous improvement [3] [4].

However, the improvement of software processes comprehend complex issues, being fundamental support them in an efficient and organized way.

This paper presents the execution of the process "Evaluation and Improvement of Process Assets", where some approaches were experimented. The process is part of the strategy in layers to define, evaluate and improve software processes, implemented on TABA Workstation.

Following this introduction, Section 2 presents the strategy in layers to define, evaluate and improve processes. Section 3 presents the execution of the proposed process and the analysis of the tested approaches. Section 4 presents the identified improvements opportunities to the process. Section 5 finally concludes the paper.

## 2. Strategy in Layers to Define, Evaluate and Improve Software Processes

In 2006, a research group of COPPE defined the Strategy in layers to define, evaluate and improve software processes to be implemented on TABA Workstation, an enterprise-oriented Process-centered Software Engineering Environment (PSEE) created to support the definition, deployment and software process improvement [5].

Nowadays (Figure 1), the strategy comprises three layers: external entity layer, organizational layer and processes execution layer.

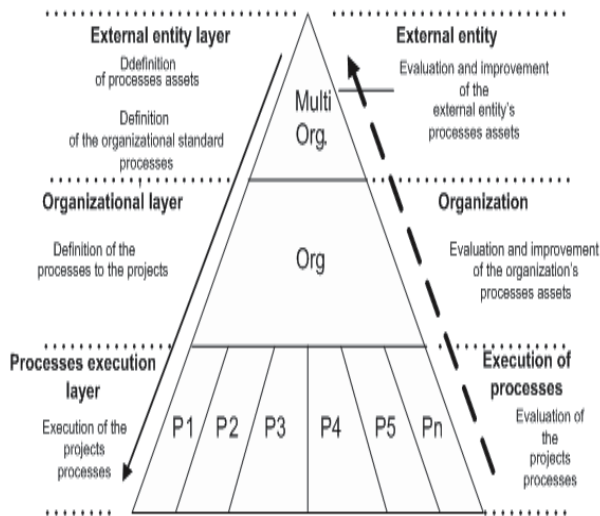


Figure 1. Strategy in layers to define, evaluate and improve software processes

According to Figure 1, in relation of process improvement, the layers interact in the way presented below:

**(i) processes execution layer and organizational layer:** a set of data from the executed processes is analyzed on the organizational layer. They can be collected from the following sources: processes adequacy evaluation (supported by Avalpro [6]); processes adherence evaluation; work products adherence evaluation; post-mortem analysis (supported by Avalpro [6]); processes monitoring indicators (supported by Metrics [7]); lessons learned (supported by Acknowledge [8]); guidelines (supported by Acknowledge [8]); processes changes rationales (supported by AdaptPro [9]); and processes changes demands.

The results of official MPS.BR [10] and SCAMPI [11] assessments can contribute with the analysis too.

**(ii) organizational layer and external entity layer:** the processes problems identified on the organizational layer, their root-causes and improvements to be implemented, besides the results of MPS.BR or CMMI assessments are sent to the external entity. These data, from many organizations, are analyzed and improvements are identified to the external entity's assets.

### 3. The process definition and execution

The approach was performed on the organizational layer and comprised the following phases: (1) Identify improvements objectives; (2) Analyze data; (3) Identify improvements; (4) Analyze and prioritize improvements; (5) Implement improvements; (6) Define preventives actions; and (7) Incorporate lessons learned.

The experience of use was in a medium size company from Rio de Janeiro between March and August of 2007. The objective of the experience was to find out evidences of viability and inadequacy aspects of the process and, specially, to experiment some define methods and techniques to support the activities execution.

**Phase 1 - Identify improvements objectives:** the purpose of this phase was to identify the improvements objectives to the organization's processes. These objectives may be reach higher levels on a maturity model (vertical improvement) or make changes on the processes to improve the productivity, the adequacy to the organization or the performance (horizontal improvement), or both of them.

The organization's directors defined hierarchically the following objectives: (i) reach the level F of the MPS.BR maturity model; (ii) improve the processes adequacy and (iii) enlarge the use of the processes in all the organization.

**Phase 2 - Analyze data:** the purpose of this phase was to identify the problems that must be solved, because are making difficult the organization achieve their improvements objectives. As the problems are organizational, it is necessary to analyze data from more than only one project.

Aiming to guide the analysis, the organization defined:

- **Business objectives:** (i) reduce the level of rework, that is an historical problem and (ii) reduce the projects costs to increase the organization's profits.
- **Product quality objectives to the organization:** (i) improve the usability, because the company develops websites; and (ii) increase the products level of reliability, reducing the quantity of defects.
- **Processes, which the data would be analyzed:** Project Management, Requirements Management and Measurement.
- **Types of problem to be considered:** (i) adequacy: related to the level of adequacy of the processes or activities ("Training Inadequacy", "Support Tools Inadequacy" and "Templates Inadequacy"); (ii)

usability: related to the difficulty to understand the description of the activities e (iii) relevance: related to the execution of not necessary activities.

- **Sources of evidence:** (i) processes adequacy evaluation; (ii) processes adherence evaluation; (iii) post-mortem analysis; (iv) processes monitoring indicators; (v) lessons learned; (vi) processes changes rationales; and (vii) processes changes demands.

On this phase we used and tested the Content Analysis Technique, analyzing qualitative data registered on the projects' sources of evidence. This methodology is usual on the Social Sciences. Its main goal is find out the more relevant subjects, reading the documents. The frequency of them defines its relevance [12].

We executed this technique to define the most relevant problems, searching and identifying the frequency which some terms were presented on the reading documents, like: "partially adequate", "inadequate", and others. Besides the frequency, we classify the terms, using others characteristics presented on the Matrix to Analysis of Problems (Table 1), created specially to support this qualitative analysis.

Table 1. Identified Improvement Opportunities

Characteristics	Description
Frequency	Frequency that a term or problem is found when we are reading the documents.
Intensity	Level of intensity (strength) of some information related to the problem.
Gravity	Level of gravity identified on the information related to the problem.
Influence on the business objectives	Level of negative influence of the information related to the problem on the defined business objectives.
Influence on the product quality objectives	Level of negative influence of the information related to the problem on the defined quality product objectives.
Impact on the adherence of the maturity models	Level of negative impact of the information related to the problem on the adherence of the maturity models.

After we consolidated the results, the following problems were selected to be considered in the actual improvement cycle: (i) Project Management: Tools Support Inadequacy and Templates Inadequacy; (ii) Requirements Management: Training Inadequacy; and (iii) Measurement: Tools Support Inadequacy.

**Phase 3 - Identify improvements:** the purpose of this phase was to identify the improvements to be implemented on the organization processes assets. For

this, we used a collaborative approach with the participation of the collaborators that somehow were involved with the processes. We decided to use and tested the collaborative approach defined on [13]. It is also used a lot on the Social Sciences.

A meeting was held with the collaborators involved on the projects (projects' managers, developers and test analyst) and the members of the software process group and metrics group.

On this meeting we tested another technique. We presented to the participants predefined cause-effect diagrams to facilitate the identification of the problems' root causes.

After the presentation of the predefined diagrams, the collaborators suggested modifications, inclusions and exclusions, generating final versions of the diagrams.

We decided to utilize the same meeting to capture the improvement opportunities, which were identified after the elaboration of the final version of a diagram. At the end of the meeting the improvement opportunities presented on Table 2 were identified.

Table 2. Identified Improvement Opportunities

Processes	Problems	Improvement Opportunities
Project Management	Support tool inadequacy	a. Deliver training on MS-Project tool. b. Integrate the TABA Workstation to MS-Project. c. Modify the TABA Workstation to a multi-user tool
	Templates inadequacy	a. Define and execute action plans only to relevant problems.
Requirements Management	Training inadequacy	a. reduce the number of types of requirements. b. improve the approach to define use cases. c. create a new activity to obtain the revision and approval of the use cases from the test analyst and the Quality Group. d. Deliver trainings on Requirements Management also to the developers.
Measurement	Support tool inadequacy	a. improve the infrastructure, specially the network's performance. b. define guidelines to help the team to collect the data. c. implement other mechanisms to increase the dissemination of the data collection culture on the organization.



At the end of the meeting we registered the following lessons learned, identified by the participants: (i) the project becomes more organized with the utilization of some artifacts; (ii) the requirements management must be performed in all the organization's projects, because it is an important factor of success; (iii) the requirements management facilitates the stakeholders' understanding of the project; and (iv) the activity spreadsheet is a very important management tool, but it should be filled exactly on the moment when the fact occurs.

**Phase 4 - Analyze e prioritize improvements:** the purpose of this phase was to analyze, prioritize and select the improvements to be implemented. Initially, to deepen the analysis, we applied a SWOT Analysis [14] aiming to test how useful this method could be to analyze the improvement opportunities. The objective was to know the factors that may facilitate or difficult the implementation of the improvements.

The strengths were the internal facilities to implement the improvement on the organization. The weaknesses were the internal difficulties. The opportunities were the external benefits obtained with the implementation of the improvements and the threats were the external benefits.

After the SWOT Analysis, we decided to test the prioritization of the improvements using a multiple criteria formal evaluation. For this, we defined the Matrix to Prioritize Improvements. Although this matrix comprises nine criteria, as we can see on Table 3, we decide to use only five: (i) urgency; (ii) impact; (iii) internal satisfaction; (iv) investment; and (v) operational simplicity.

Table 3. Criteria of the Matrix to Prioritize improvements

Criteria	Criteria Definition
<b>Seriousness</b>	Seriousness or damage to the organization if the improvement will not be implemented.
<b>Urgency</b>	Urgency to implement the improvement.
<b>Tendency</b>	Probability of the process performance gets worse if the improvement will not be implemented.
<b>Impact</b>	Impact of the improvement on the process performance.
<b>Internal satisfaction</b>	Collaborators satisfaction with the process if the improvement will be implemented.
<b>Investment</b>	Needs of resources (financial, human and technological) to implement the improvement.
<b>Time</b>	Necessary time to implement the improvement
<b>External satisfaction</b>	Clients' satisfaction with the process if the improvement will be implemented
<b>Operational simplicity</b>	Simplicity to perform the necessary actions to implement the improvement.

As we can see, each criterion was evaluated using a scale comprised of the following values 3, 5 and 7. The improvement that obtained the highest level of priority was: "Create a new activity to obtain the revision and approval of the use cases from the test analyst and the Quality Group".

**Phase 5 - Implement improvements:** the purpose of this phase was to implement and institutionalize the improvements selected on the anterior phase, including the planning, execution and evaluation of pilot projects. As this phase had already been evaluated by COPPE in experiences of processes implementation (consultancy) [15], it was not evaluated, but a report containing the results was elaborated to be sent to the external entity.

**Phase 6 - Define preventive actions:** the purpose of this phase is to define actions that may prevent the organization against imminent problems. For this, we decide to know the relations of the causes of the problems. We chose to test the utilization of the Matrix to Find-out Relationships, adapted from the Matrix to Discovery, suggested by Bacon and presented in [16]. The utilization of this matrix aimed to facilitate the identification of the relationships of the causes identified on the meeting hold on phase 3 and their strength of influence (relationship coefficient). Besides, aiming to improve the visualization, we tested the utilization of circles to represent the strength of influences among the causes, similarly in the Matrix of Distances, illustrated in [17]. It aimed, mainly, to facilitate the identification of influence zones.

**Phase 7 - Incorporate lessons learned:** the purpose of this phase was to register the lessons learned captured during the execution of the phases, aiming to be reused in future situations. At this experience, many lessons learned were captured and registered.

#### 4. Analysis of the Results

After the execution of the proposed process, we analyzed the main utilized methods and techniques to identify their weaknesses and strengths to support the activities execution and to suggest some improvements. Table 4 presents the results obtained by this analysis.

Table 4. Analysis results

Technique/Method	Results
<b>Content Analysis</b>	The utilization of this technique helped a lot to organize the analysis of the qualitative data. However, it is important to support this technique with another

	method, like coding, to guide better the identification of similar piece of text.
<b>Collaborative Approach</b>	The engagement of the collaborators was enhanced and they could better condition to identify the problems' root causes and suggest adequate improvements. Besides, it permitted to increase the knowledge of the collaborators on the organization's processes. This approach also permitted the exercise of critical analysis, that increases the teams' level of reflection and the socialization of the knowledges [18].
<b>Predefined cause-effect diagram</b>	The use of pre-defined diagrams helped the participants to remember their experiences with the processes. They could be seen as a powerful catalyst to the emergence of new causes. However, it can be also useful to identify the relations between the causes.
<b>SWOT Analysis</b>	The application of the SWOT Analysis, before the prioritization of the improvement opportunities was very important, because it permitted the analyst to deepen their knowledge in relation of the improvements. Were considered others important factors.
<b>Multiple criteria formal evaluation</b>	This approach was very useful, because we could see the difference when we decide only using feelings when we decide using well-defined criteria. Moreover, it helped a lot the decision-maker.
<b>Causes relationship</b>	It was not adequate to identify imminent problems. Maybe it was more relevant to support the identification of improvement opportunities. However, we could observe that the utilization of circles to represent the strength of the relationships was very useful to identify zones of influence.

## 5. Conclusion and Future Works

The results of the first experience demonstrated that the approach was effective, supporting the organization to identify and prioritize improvements. The phases guide the execution of the approach adequately, providing knowledge that helps the collaborators. Besides, the methods and techniques were almost always very useful and relevant to guarantee the efficiency of the activities execution.

After this experience, we may explore the following further works: (i) Plan and perform a formal case study to validate this process; (ii) Test others techniques and methods; and (iii) Develop a tool on the TABA Workstation to support this process.

## 6. References

[1] Araujo, E. E., R., Meira, S. R. L., "Competitive Insertion of Brazil on the International Market of Software", [http://www.softex.br/portal/\\_publicacoes/publicacao.asp?id=806](http://www.softex.br/portal/_publicacoes/publicacao.asp?id=806)

[2] ISO/IEC, ISO/IEC 25000: Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE”, 2005.

[3] SOFTEX, Brazilian Software Process Improvement – General Guide version 1.2, available at <http://www.softex.br/mpsbr>.

[4] CMU/SEI, CMMI for Development version 1.2., CMU/SEI-2006-TR-008, 2006.

[5] Montoni, M. et al., “Enterprise-Oriented Software Development Environments to Support Software Products and Processes Quality Improvement”, In: Proceedings of PROFES 2005, 2005, pp. 370-379, Oslo, June.

[6] Andrade, J. M. S., Evaluation of Software Process in ADSOrg, Dissertation of M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brazil, 2005.

[7] Schnaider, L. et al., “MedPlan: an approach to measurement and analysis on projects of software development”, In: Proceedings of the SBQS 2004, 2004, Brasília.

[8] Montoni, M., Knowledge Acquisition: an application on the software process, Dissertation of M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brazil, 2003.

[9] Berger, P., Instantiation of Software Process on Configured Environment in the TABA Workstation, Dissertation of M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brazil, 2003.

[10] SOFTEX, Brazilian Software Process Improvement – General Guide version 1.2, available at <http://www.softex.br/mpsbr>.

[11] CMU/SEI, CMMI for Development version 1.2., CMU/SEI-2006-TR-008, 2006.

[12] Bardin, L., 1977, Content Analysis, Lisboa, Edições 70.

[13] Cordioli, S., “Participation Approach”, In: BROSE, Markus (org), Participation Methodology: an introduction to 29 tools, Tomo Editorial, 2001, pp. 25-46.

[14] Keok, C. B., “Work in progress - integrating bos, swot analysis, balanced scorecard and outcome-based framework for strategy formulation of engineering school”, In: Proceedings of FIE '07, Milwaukee, 2007, October.

[15] Ferreira, A. I. F. et al., “ISO 9001:2000, MPS.BR Level F e CMMI Level 3: a software process improvement strategy on company BL”, In: Proceedings of SBQS 2006, 2006, pp. 375-382, Vila Velha.

[16] Moles, A., The Scientific Creation, São Paulo, Perspectiva Press, 1971.

[17] Moles, A., The. Sciences of the Imprecise, Rio de Janeiro, Civilização Brasileira Press, 1995.

[18] Popper, K.R., The Mith of the Framework, Lisboa, Editions 70, 1996.

# Establish Decision Making Process for selecting Outsourcing Company

Akihiro HAYSHI

R&CM, Capacity Management, IBM Japan,  
19-21, Nihonbashi, Hakozaiki-cho, Chuo-ku, Tokyo, 103-8510,  
E-mail: AikPIXY@jp.ibm.com

**Abstract** *Outsourcing is prevalent among system development activities. According to relevant statistic source, outsourcing is not always successful from the viewpoint of cost-performance, despite outsourcing merit being emphasized. This is considered because that decision-making process to select outsourcing company is not established. In this paper, we adapt the framework of CMMI and traditional decision-making process such as Kepner-Tregoe Method as the basis, and attempt to establish reliable decision-making process by combining AHP method and Even-Swap method. Also, with concept of liner programming, we verify validity of final alternatives derived by the process*

## 1. Introduction

Recently, there is increasing number of cases to outsource some part of system development project to an external organization. Outsourcing companies are not limited to domestic enterprises, but offshoring to enterprises in emerging countries such as China and India are also increasing.

The rationales for outsourcing to the external organization include; to focus on the core competence; to pursue cost advantage; to shorten lead-time of product development; and utilization of asset of external enterprise.

Despite outsourcing of system development project is promoted to achieve cost advantage etc, there are number of troubles such as delivery delay or quality issue due to skill-shortage and lack of communication capability in outsourcing companies have occurred. Even after successful project delivery with selected outsourcing company, if you kept ordering to the same organization for four years, your cost reduction effect often decreases [1].

About 65% of domestic enterprises have been selecting their outsourcing companies by criteria such as "past associated organization or introduction by the customer" [2]. There are some examples of having selected outsourcing companies by applying original purchase management process, as with purchase of a PC, but there are few cases that the Decision Making process for selecting outsourcing companies is established.

Even with reports at international conference on process improvement such as SEPG, or domestic workshop such as JASPIC and SPES in Japan, reliable and easy to use decision making process for selecting outsourcing companies is not yet established.

From these background, establishment of the decision making process for selecting outsourcing companies in system development projects is taken up as the theme of this research.

In this paper, we first prepare the criteria and the parameter for selecting external outsourcing companies in system development projects. It aims to establish the decision making process that is reliable by combining a simple decision making technique to traditional decision making process as the basis. Then, the hypothetical criteria and the evaluation results are verified by reference to the idea of linear programming.

As an early research on outsourcing of IT Management, Lacity[3] pointed out that the success condition of the outsourcing of IT management depends on the maximization of flexibility and the control ability of the system and discussed a concrete selection method for selecting outsourcing companies and their estimate.

Huber[4] listed various conditions of selecting outsourcing companies which change high fixed costs into the additional value ahead, then discussed a method of concentration of candidate organizations. Cross[5] reported on the means to evaluate the candidate company list of outsourcing companies which introduce the principle of competition for IT outsourcing strategy.

However, these reports are discussions on the idea of the strategic outsourcing. Method for selecting one company by considering criteria such as geographical scope, offer cost, services that can be provided, and technical feature after several candidate-outsourcing companies is decided was not presented.

There is no previous research that objectively evaluated the decision making process for selecting outsourcing companies.

The composition of this paper is as follows. In Chapter 2, the decision making process of CMMI and issues to be solved are described. In Chapter 3, the principle for using traditional decision making process and the assumption is

described. In Chapter 4, in order to solve the issues described in Chapter 2, by using a traditional decision making theory as a basic framework and various decision-making processes in addition, method for correctly weighting evaluation criteria and evaluation of alternatives is proposed. In Chapter 5 the issue presented in Chapter 2 is verified. In Chapter 6 discussion is made, and the conclusion is given in Chapter 7.

## 2. The Issue in Selecting Outsourcing Companies

### 2.1. Limited capability of using CMMI for Selecting Outsourcing Companies

Process evaluation model CMMI Ver 1.2[6] developed in the United States is often used in system development projects. In CMMI, 'Decision analysis and Resolution' (DAR) is described as a process of decision-making process. Figure 1 shows six activities of DAR.

SP 1.1 Establish Guidelines for Decision Analysis  
 SP 1.2 Establish Evaluation Criteria  
 SP 1.3 Identify Alternative Solutions  
 SP 1.4 Select Evaluation Methods  
 SP 1.5 Evaluate Alternatives  
 SP 1.6 Select Solutions

**Fig. 1 DAR process of CMMI**

When selecting outsourcing companies according to the steps in Figure 1, the organization establishes guideline such as "Select enterprise that is balanced on cost and skill", and establishes the evaluation criteria such as "Cost, schedule, skill, and past dealing results", and specify alternatives such as "Company-A, Company-B, and Company-C".

Now, even if the candidate outsourcing companies are selected, SP1.4 "Select Evaluation Methods" is often not implemented efficiently enough. According to CMMI, "Evaluation method" includes simulation model, probability model and decision-making.[6] However, no concrete procedures are described.

It is very difficult and unrealistic for the project managers of system development projects to investigate advanced expertise of simulation model, probability model and decision-making model and establish the application for selecting outsourcing companies within tight development schedule.

### 2.2. The issue to be solved and requirement

Organizations where CMMI process improvement best practice model has been introduced, have issue to establish the decision making process as the method for selecting the outsourcing companies.

The decision-making process to be established has two requirements.

#### (1) Simple method that project manager of general knowledge level can use

The decision making process is an indirect activity of the project. It is preferable to complete within short time. Method must not require one weeklong discussion only for selecting outsourcing companies. The simple method that general skilled project managers who doesn't have advanced expertise of simulation model, probability model and decision-making can use is preferred.

#### (2) Reproducibility of result when executed by two people in charge with understanding of circumstances of the project.

As the method of the decision-making, association, or selection might introduce outsourcing companies without valid rationale such as wild guess or flashes. If you select the outsourcing companies without rationales, it is difficult to have reproducibility.

Process is required to give same result when executed by two people in charge of the project such as a project manager and a member from outsourcing company.

## 3. Reused Framework of Traditional Decision Making

A traditional method of the decision-making is used in this research to establish the outsourcing companies selection decision-making process. Traditional decision-making process consists of four steps, which is Situation Recognition, Trouble Shooting, Decision Making, and Risk Management. For example, Kepner-Tregoe Method [7] of the United States (hereafter, abbreviated as the KT method), EM method and THP method (both roots as the KT method) are assumed as traditional methods of the decision-making.

Traditional Decision-Making Process is executed by following steps:

- 1) Define the issue that need to make decision,
- 2) Set up the evaluation criteria for the alternatives,
- 3) Set relative weight of the evaluation criteria,
- 4) Enumerate all alternatives,
- 5) Evaluate alternatives based on the criteria,
- 6) Evaluate value combined Weight of criteria and each value
- 7) Select the highest score of the comprehensive evaluation.

Because a total score is already calculated in this method when a final solution was selected, a quantitative judgment without intuition and the experience but the numerical result can be done. This method is a decision making process used most traditionally, and the easily used methodology.

However, there are three preconditionsto apply a traditional decision making methodology to the method of selecting the outsourcing companies in the system development projects [8]

- 1) All criteria of selecting the outsourcing company can be recognized and the accurate weights of criteria are known.
- 2) All alternatives are recognized and unnecessary alternatives can be excluded.
- 3) Alternatives can be evaluated objectively and numerically.

However, it is often not the case that above preconditions are fully satisfied and therefore a traditional method of the decision-making is not reproducible enough.

#### 4. Proposal of the Decision-Making Process of Selecting Outsourcing Companies

In this chapter, the decision making process that is reliable, is established by using the framework of the traditional decision making method. The assumption requirement pointed out in Chapter 3 is met by the hierarchical analysis and Even-Swap method. The process consists of the following three phases.

Phase1) Set evaluation criteria, define absolute condition and relative condition, and set weight on relative condition by the hierarchical analysis method.

Phase2) The selected candidates are narrowed down to about ten organizations by using the Even-Swap Method.

Phase3) Alternatives are evaluated against absolute condition and relative condition, then results are collated using linear programming approach.

Based on the framework of the traditional decision making theory, three phases proposed are shown in Figure 2. Detailed explanations are in following Phases.

##### Phase 1: The Hierarchical Analysis Method is applied to Weighted Criteria

In Phase 1, the selection criteria are classified into absolute condition and a relative condition, and weight is set between criteria of relative conditions.

The absolute condition is a mandatory condition such that must be met, and outsourcing companies, which fail to meet this condition, will not be selected.

For example, "The order budget to outsourcing companies is to be kept less than 10 million yen" becomes an absolute condition in a limited project budget.

On the other hand, a relative condition is such that it's preferred to be met, but not an absolute condition. For example, use C++ for development language, achieved CMMI level 3. Because the importance is uniformly different between relative conditions, weight is set to them.

The criteria in traditional decision-making method are usually evaluated by ten stages evaluation where score of 1-10 is set on each of the relative conditions (There are five stage evaluation method, comparative assessment method).

Now, which relative condition to be considered the most important and should it be set at 10 points? How can you adjust other relative conditions to 7 points or 8 points compared with most important relative conditions? If the point is 7, why isn't it 8, why not 6, but 7? The rationale of the logical grounds is extremely vague. It might be difficult to reproduce the same point accurately, even when the same evaluator executes it for the second time.

In this paper, by using the Analytic Hierarchy Process[9] (hereafter abbreviated AHP), relative conditions are weighted. AHP is a technique for choosing the best evaluation by synthesizing a relative importance of the element in each hierarchy after arranging them to a layered structure of the target, evaluation criteria, and alternative approach, when there are multiple criteria. In this research, evaluation criteria are weighted using AHP.

For calculation of weight of AHP, there are an eigenvalue method and a geometric mean method. The eigenvalue method is excellent in the point of best approximation process. The geometric mean method is excellent in the point of ease of calculation. In this paper, the geometric mean method is used, valuing its simplicity.

In geometric mean method, weight is set at an average value of a pairwise comparison. For example, with criteria "Price", "Years of experience", "Deal result", "Presence of the contract for maintenance", and "Distance between the order side", point is set from the classification of (1) Same, (3) a little, (5) rather, (7) plentifully, (9) Absolutely (inverse is used when you calculate opposite way). The product of the evaluation result of the criteria is calculated, find their geometric mean, and the proportion of each geometric mean in the total of the geometric mean is considered to be the weight.

With traditional decision making method, usually integer value is used for weight (refer to Figure 3), where thousandth value is used in AHP weight.

Phase 2 is a phase that narrows down the list of outsourcing companies to about ten companies or less that can be selected realistically.

When CMMI is used for process management, candidate of outsourcing companies is predefined as "Sup-

Absolute Conditions		Alternative 1		Com A	
	Phase 2	Evaluation Data		Clear	
Cost < 12 million yen		13 Million		✓	
SAP > 5 Years Experience		SAP 5 Years		X	
Relative Conditions	W.	Evaluation Data		P	S
Deal Results	5	5 Years		1	5
Maintenance	4	Yes		1	4
D	Phase 1	3	Phase 3	1	3
Total Points					12

Fig. 2 Decision Making Process for Selecting Outsourcing Company

plier's candidate's list" or "List of desirable supplier" [6] in Supplier Agreement Management process. The number might reach several dozen of companies. If all of these several dozen companies are to be scored as candidates, it will not result well-modulated selection. It is preferable to exclude unnecessary alternatives beforehand.

	Price	Yer	Dea	Main	Dist.	Calculations
Price	1	3	5	7	9	$1*3*5*7*9$
Years	1/3	1	1/3	1/3	5	$1/3*1*1/3*1/3*5$
Deal	1/5	3	1	7	1/5	$1/5*3*1*7*1/5$
Maint.	1/7	3	1/7	1	3	$1/7*3*1/7*1*3$
Dist.	1/9	1/5	5	1/3	1	$1/9*1/5*5*1/3*1$

	Product	GeoMean	Weight	Description
Cnt.	945.000	3.936	0.575	$\Leftarrow 3.926/6.846$
	0.185	0.714	0.104	
	0.840	0.966	0.141	
	0.184	0.713	0.104	
	0.037	0.517	0.076	
	Total	6.846	1.000	$\Leftarrow$ Weight Total 1

Fig. 3 Criteria Prioritization by AHP

**Phase 2: Applying Even-Swap Method to Narrow Down Alternatives**

In this paper, the Even-Swap method [11] (ES method) is used to narrow down alternatives. The ES method is a way that normalizes value of result of various alternatives by adjustment, and omits their effect upon selection.

For example, assume there are two criteria "price" and "years of experience". Then assume Company-A price is one million yen higher than Company-B and Company-A's experience is two years longer than Company-B. The judgment is affected by the consideration whether to prioritize the five years of experience of Company-A is higher price, or prioritize lower price even if Company-B has only three years of experience.

Now, assume that know-how of one-year experience can supplement the price difference of one million yen. That is, cost advantage of one million yen by selecting Company-B is counterbalanced after one year. The price 12 million yen with four years experiences has equal value to the price 11 million yen with three years experience after adjustment.

	Alternatives	
	Company-A	Company-B
Price	1200	1100
Exp. Years	5	3

Fig. 4 Result Table

	Alternatives	
	Company-A	Company-B
Price	1200	<del>1100</del> 1200
Exp. Years	5	<del>3</del> 4

Fig. 5 Applied Even Swap to Alternatives

The price of the two companies is 12 million yen and the experience years are five years for Company-A and four years for Company-B in Figure 5. Company-B has become disadvantageous to Company-A, and so there is no reason to keep Company-B as an alternative. Company-B can be excluded from alternatives candidates.

Unnecessary alternatives are excluded by repeating this procedure for narrowing down disadvantageous alternative.

However, the rationale of illustrated Even Swap example "One year of experience can counterbalance one million yen of proposal price" is not very strict. Moreover, it is nonsense to have an Even-Swap such as "The difference of the distance 10,000km with the order side counterbalances proposal price of one million yen" when you are selecting offshore organization like India.

So, in this paper, the ES method is only used within the scope where the rule of thumb built up at the order side such as prices and years of experience. The alternatives are not narrowed down until the last one by the ES method, but narrowing down to about ten organizations.

Also, with case shown in Figure 5, you can either remove the price (criteria) from the examination item of the decision making since it is already equal, or you can remove Company-B (alternatives) as Company-B is completely disadvantageous to Company-A. In this paper, not the criteria but alternatives are narrowed down according to the investigation purpose.

**Phase 3: Evaluate and Verify Alternatives by Linear Programming.**

In Phase 3, alternatives are evaluated by criteria and an integrated point is calculated. Then the result is verified by linear programming.

There are absolute conditions and relative conditions in the basis of selection as shown in Figure 2. Since the alternatives won't be selected unless absolute conditions are satisfied, absolute conditions are initially judged by  $\checkmark$ X. If at least one absolute condition is evaluated X, no more evaluation will be done.

Then, relative conditions of the alternatives that its absolute conditions are already evaluated  $\checkmark$  are evaluated. In the evaluation of the relative conditions, the alternatives narrowed here are compared and 10 points is given to the alternative that best meet a relative condition. Next, 1-9 point is set to other alternatives by the comparison with alternatives with 10 points. Therefore, 10 points is given to one alternative and so are 1-9 points to other alternatives.

Afterwards, alternatives with the highest integrated point calculated by the product of weight of the criteria and evaluation result is selected.

However, as pointed out in Chapter 1, available alternatives of the outsourcing companies in system develop-

ment projects are often introduced by past association or the customers. Sometimes the candidate alternatives might be a small business where there is possibility of bankrupt by cash flow if the company cannot receive an order. Or the selection candidate might be a sole proprietorship that senior retired employee had founded.

It is difficult to objectively evaluate the alternatives numerically due to cognitive biases. The cognitive bias is the phenomenon that evaluation of certain object is dragged by a remarkable feature, or that evaluation is misinterpreted by influence from specific information or memory [12].

For example, Halo Effect that the evaluation is unnecessarily improved by the introduction of the customer, Ranking Inflation that makes the evaluation lenient when the candidate has long term association, or Ranking Compression that the evaluation becomes noncommittal near center saying "It cannot be said either".

So, in this paper, objectivity of evaluation is verified. The 1st and 2nd place of the result by 10 points evaluation are verified using the idea of the linear programming. Linear programming is a technique used to calculate the best resource allocation for production management and operations research. It is used to calculate the optimum distribution that the targeted value becomes the maximum in the limited condition of the linear expression.

Evaluation point required for the 2<sup>nd</sup> place alternative to exceed 1<sup>st</sup> place alternative's point is calculated by linear programming method. This value is then compared to the current scores of alternatives, and with consent by more than two people, evaluation is considered final. If not, then alternatives are reevaluated.

For example, as shown in Figure 6, assume that evaluation result of alternative A is 7 point, and of alternative B is 9 point when the final evaluation is completed. In this case, B is finally selected. However, if the evaluation result of alternative A is 10 point, then the final alternative become not B but A. Then, the evaluation value of a relative condition that an integrated point becomes 10 point is calculated using the linear programming.

Con	Wei	Alt.A		Alt.B		Lin.Prog.	
		Sco	Poit	Sco	Poit	Sco	Poit
1	0.345	10	3.45	10	3.45	10	3.45
2	0.531	3	1.59	5	2.65	6	3.19
3	0.123	5	0.62	5	0.62	5	0.62
4	0.51	3	1.53	5	2.55	6	3.06
	Total	7		9		10	

**Fig.6 Verification using Linear-Programming**

The linear programming is built in as one of the add-in functions of MS-Excel normally used in current IT companies. With required limiting condition and targeted values, the best search result will be returned. In the right-

most column of Figure 6, values required for the overall judgments to become 10 points are shown for each condition, using the linear programming. By comparison, you can see that Alternative A exceeds B if relative conditions 2 and 4 were 6 points respectively.

Now, whether cognitive biases of "Negative Leniency" effect that the evaluation of A company lowers unnecessarily had occurred is verified by investigating the person in charge of the evaluation of relative condition 2 and 4, content of the relative conditions, and A company.

For instance, there is a possibility to generate Cognitive biases due to the person in charge of the evaluation with insufficient understanding of neither relative condition 2 or 4, or past incident of delivery delay and/or a quality trouble by Company-A. In such cases, person in charge of the evaluation need be changed and/or mean value of two or more evaluation results should be taken as the measures of the adoption.

If the result remains the same after person in charge of selecting outsourcing company is changed, then Company-B is selected as outsourcing companies as shown in Figure 6. If the result differs, then process in phase 3 is repeated and final outsourcing company is selected.

## 5. Verification

In this paper, the establishment of the decision making process for outsourcing company selection is taken up as main theme and the requirement for the solution was presented in 2.2.

In Chapter 5, whether the method proposed in this paper meet the requirement is verified.

### (1) Simple method that project manager of general knowledge level can use.

All the proposed techniques in this paper, which are KT methods, AHP, and ES method, can be implemented on MS-Excel. For more complex calculations for AHP, which are pairwise comparison, geometric mean, and eigenvalue (unused in this paper) can be done using commercially available materials and free software. User can semi-automatically obtain selection result by only judging weighting. Knowledge of advanced mathematics in complex eigenvalue or geometric average calculation is not required.

Moreover, by implementing on the spreadsheet tool, a more accurate evaluation can be made by referring to the knowledge from past projects reflecting what result actually became.

As a result, even a project manager who doesn't have advanced knowledge of the theory of probability and simulation can obtain a proper decision making result by use of the technique proposed in this paper. It can be said that the method proposed here can be use by the project manager of a general knowledge level.

## (2) Reproducibility of result when executed by two people in charge with understanding of circumstances of the project.

The technique for the selection of the outsourcing companies has mainly three points; weight setting to the criteria; narrowing down alternatives; and evaluation of alternatives by referring to the absolute conditions.

In this paper, AHP was used for the weight setting of the criteria. AHP is a well-known method that quantifies person's sensuous evaluation which has already been confirmed as reliable method.

The ES method was used for narrowing down alternatives. There is essentially reproducibility as long as the trade-off condition becomes clear, since ES method excludes unnecessary alternatives by setting the trade-off condition such as costs and years of experience.

In the evaluation of alternatives against the absolute conditions, most suitable alternative was evaluated to a relative each condition as 10 points, and other alternatives were evaluated relatively. In addition, alternatives in 1st and 2nd place were verified by calculating value required for the 2<sup>nd</sup> place alternative to exceed value of 1<sup>st</sup> place alternative using linear programming. It can be expected that this process to yield same result almost every time as long as two or more people discusses along the procedure.

Therefore, it is judged that the selection result has reproducibility from the technique proposed in this paper if two people who know circumstances of the project execute the method.

## 6. Discussion

In this paper, five decision-making methods are used. They are CMMI, a traditional decision making methods like KT Method, AHP methods, ES methods, and linear programming. These techniques are independent decision making techniques respectively, and any of these decision-making method can be solely used for decision-making.

For example, AHP that was used Chapter 4 can be executed to the selection of the final alternative by evaluating not only setting weight but evaluate the criteria.

However, there is some occasion that selection should be made amongst unknown organizations where there is no past dealing experience. If the AHP is to be consistently used, you will be comparing whether unknown Company-A or an unknown Company-B is better, which is not very wise idea.

Because five decision-making techniques used in this paper have both merits and demerits like this, the proposed method supplements insufficient area of one method by another, and aims to achieve more reliable decision-making process.

## 7. Conclusion

Decision making process for the selection of outsourcing companies is proposed in this paper by recognition that it was a situation in which the selection of the optimal outsourcing companies was not necessarily done appropriately, considering increasing requirement of outsourcing in IT system development.

The proposal in this paper uses the frame of traditional decision-making process KT method with CMMI, the best practice of the system development, in its basis. AHP is used for the weight setting of the criteria, ES method is used for the trade-off of alternatives, and, in addition, the process became more reliable by using the idea of the linear programming for the selection of result.

By applying the technique in this paper, evaluation that has been made intuitively can now be supplemented with the method of the decision-making by which effectiveness is confirmed. In addition, the idea of the linear programming verifies the execution result and the reliability of the evaluation result is improved.

In this paper, a theoretical frame in the decision making process is proposed. However, application evaluation in an actual project is not yet conducted. It is future tasks to improve this proposal continuously by applying to an actual project and evaluating it.

## 8. Reference

- [1] <http://www.ciojp.com/contents/?id=00003744;t=12>
- [2] White Paper of Japan Information Service Association 1996,1997
- [3] Mary C. Lacity, Leslie P. Willcocks, David F. Feeny, "IT Outsourcing: Maximize Flexibility and Control", Harvard Business Review Article, May 1, 1995
- [4] Richard L. Huber, "How Continental Bank Outsourced Its "Crown Jewels", Harvard Business Review Article, Jan 1, 1993
- [5] John Cross, "IT Outsourcing: British Petroleum's Competitive Approach", Harvard Business Review Article, May 1, 1995
- [6] <http://www.sei.cmu.edu/cmmi/translations/japanesemodels/index.html>
- [7] Charles H. Kepner, Benjamin B. Tregoe, "The New Rational Manager", Princeton Research Press, 1981
- [8] Ichiro Innami, Sugureta Ishikettei, Chuokouron, 1997
- [9] T. L. Saaty, The Analytic Hierarchy Process. McGraw-Hill, NewYork, 1980.
- [10] Hideji Takeda, Saaty no houhou niyoru Weight no Jakkan no ginmi, AHP Jireishu, JUSE, pp.223-246, 1990
- [11] John S. Hammond III, Ralph L. Keeney, Howard Raiffa, "Even Swaps: A Rational Method for Making Trade-Offs", Harvard Business Review Article, Mar 1, 1998
- [12] Globis Business School, MBA Glossary, <http://gms.gl>



# From Strategy to Solution: A Lightweight Semi-Prescriptive Approach for Software Development Lifecycle with Outsourcing Support

Nelio Alves  
Federal Institute of  
Triângulo Mineiro  
nelio@iftriangulo.edu.br

Sergio Paim  
Invit Information Services  
sergio.paim@invit.com.br

Alexandre Cardoso  
Federal University of Uberlândia  
alexandre@ufu.br

Edgard Lamounier  
Federal University of Uberlândia  
lamounier@ufu.br

**Abstract-** This paper presents an ongoing academic-enterprise collaborative research work on specifying and implementing a lightweight semi-prescriptive software development lifecycle method with hybrid agile-prescriptive features. This method proposes features and design choices that are characterized and discussed. For instance, we choose formal requirement management with proper change traceability, as well as prescriptive software architecture in order to provide advantages like team scalability, reliable requirement realization and scoping. We introduce the concept of *implementation scenario*: a particular subset of requirement – even non-functional – that greatly improves risk management and overall architecture stabilization efficiency, and also is one of the key aspects for agile-prescriptive integration. The method also proposes a built-in support to software development outsourcing, which consists of a process-centered interface between customer and provider based on activity assignment, artifact custody and the method design itself. The method has been informally tested on small and medium size projects so far. Tests have shown significant results that are presented and discussed.

## I. INTRODUCTION AND MOTIVATION

A recent substantial process-centered survey on software development methodology [1] analyzed the state of the art of this research field by examining seminal, disciplined and agile methods. Among their conclusions, we discuss those that serve as part of the foundation to our work motivation:

- **Integration needs:** According to their analysis, disciplined methods and their agile counterparts have no other choice but to converge. Another source of inspiration to this matter is the work published in [2].
- **Methodological neglecting:** Software development methods are usually built without considering a proper methodology foundation or systematic approach.
- **Agile approaches common problems:** They concluded that, despite remarkable achievements, agile methods are still not mature enough [2, 3, 4, 5] and the problems more commonly cited are lack of scalability, unrealistic assumptions and lack of a specific, unambiguous process.

We also point out some other issues that motivate this work:

- **Software outsourcing:** The whole mentioned survey [1] does not even mention outsourcing. We could observe some lack of process-centered methodological approaches to support outsourcing. In other words,

software development processes usually do not include specific support to outsourcing.

- **Requirements:** Formal requirement management is crucial to ensure proper communication and agreement between customer and provider. In [1], Requirements Engineering is considered a weak link in many methods. We also observed it in our experience and decided that a formal approach with proper change traceability to requirements is mandatory to the purpose of this work.

In order to address the mentioned issues, this paper presents a software development lifecycle method and discusses its general and specific features that cover those issues.

An experienced IT company is playing a core role on this research as it has been given lots of contributions on the process specification and testing. The method has been tested on small and medium size projects and tests have shown significant results, mainly on efficiency and risk management.

This paper is considered a proposal because quantitative empirical validation was not yet carried out.

## II. RELATED WORK

We considered the core related work about this research two-fold: those about (1) software development methodology and those about (2) outsourcing software.

There are many approaches to disciplined and agile software development lifecycle methodology. This brief review focused on those which have influenced this work. More information about software methodology can be found in [1].

Rational Unified Process (RUP) [6, 7] and its nonproprietary counterpart USDP [8] was the major theoretical and practical basis for this work. Besides, it is widely experienced for more than six years by the company where this work is being applied. RUP is a use-case driven, architecture centric, interactive and incremental method. RUP uses the Unified Modeling Language (UML) [9] as its default modeling language. RUP consists of a huge specification and it is able to customization in order to facilitate manageability. An extended variant of RUP called Enterprise Unified Process [10] was later proposed.

Scrum [11] is a framework for software development first presented in [12] focused on strong team interaction and structuring the development disciplines in 2-4 week iterations called sprints. This framework consists of three cyclic phases: pre-game (planning, high level design, architecture), development (sprint execution) and post-game (integration and

delivering activities). Scrum is usually used in combination with another method – typically an agile one – and prescribes some practices like doing daily meetings and keeping a physical visible task board with work orders and a release burndown chart. Our approach incorporated the task board prescribed by Scrum as part of one of the management activities, which has shown very effective.

Much research has been carried out on information systems outsourcing, but few treated software development outsourcing specifically [13]. We found more publications about the broader field of IT outsourcing and even more about general cross-organization workflow. Several outsourcing topics often appear like its adaptive nature [14], success factors [15], case studies [16] and guidelines [17].

We could not find any methodological approach to software development outsourcing closely similar to our proposal and the literature analysis shows that there is still a lot to do on process-centered methodological approaches to this subject.

### III. METHOD OVERVIEW

Our software development lifecycle method was defined in terms of seven IT processes, grouped according to Service Design and Service Transition functional areas from the service structure of ITIL - a widely accepted library of good practices for IT service management [18]. Table I shows those seven processes and grouping.

TABLE I  
IT processes for Service Design and Service Transition

Service Design	Service Transition
<ul style="list-style-type: none"> <li>• Service Inception</li> <li>• Process Engineering</li> <li>• Requirements Engineering</li> </ul>	<ul style="list-style-type: none"> <li>• Improvement Management</li> <li>• Improvement Engineering</li> <li>• Improvement Validation</li> <li>• Improvement Deployment</li> </ul>

A brief description of each IT process responsibility is shown below. In next section we give more details about the process specification pattern.

**Service Inception:** this process first takes strategic guidance and updates service portfolio by identifying and describing an IT service that assist an automation demand being analyzed. It then calls the Requirements Engineering process to identify initial scope of an application solution for the service and performs functional and economical feasibility analysis, structuring initiatives for solution development if feasibility is confirmed. If necessary, this process is previously assisted by the Process Engineering one for business modeling and exploration of automation demands. The Service Inception purpose is similar to the one from Inception phase from RUP.

**Process Engineering:** responsible for business modeling with strategic alignment and continual optimization. More details are discussed on next section.

**Requirements Engineering:** responsible for performing one requirement cycle. This cycle consists of elicitation and specification of automation requirements (without violating architectures integrity), formal change traceability

management, scope estimation and formal costumer-provider validation.

**Improvement Management:** responsible for performing project management activities during Service Transition stage.

**Improvement Engineering:** responsible for performing one development cycle, which consists of activities related to architecture refinement, design, implementation, tests, integration and production of supporting material. This process has an agile fashion and more details are given on Section V.

**Improvement Validation:** responsible for performing one validation cycle, which consists on activities related to deployment to validation environment and validation of the application software and its supported IT services.

**Improvement Deployment:** responsible for performing activities necessary to deploy the new solution to production (as IT services), like defining infrastructure resources, migration plan execution, stability testing and user training.

### IV. USED METHODOLOGY

All seven IT processes were modeled according to a well-defined Process Engineering foundation. In fact, the method we used was exactly the “Process Engineering” process we have created (see Table I). Note that this is a meta-process as it is a business process that specifies how to specify business processes (where IT process is nothing but a specialization).

Our “Process Engineering” process was highly inspired by the Business Modeling discipline of RUP, where some aspects were simplified and others added.

We made a detailed textual specification complemented by graphical representations for each IT process and its activities. For space constraints, we only present the “Process Engineering” process’ behavioral and structural models in Figures 1 and 2. Notice that we used UML Activity Diagram and UML Class Diagram with proper use of stereotypes.

### V. FEATURES AND DESIGN CHOICES

In this section we discuss main design choices we made and point out reasons that have lead to them. This discussion is then complemented in next section, where we present outsourcing-related decisions.

**Formal requirement management:** the Requirement Engineering process was defined in order to accommodate an outsourcing requirement logistics. On every requirement cycle (including the first), a Specification Change document is produced and all related requirements are baselined. After customer-provider mutual requirement elucidation and specification, requirements are formal validated by customer and then are baselined by the customer’s Project Manager and communicated on both sides with contractual value. This methodological structure ensures proper requirement management and change traceability.

**Prescriptive architecture:** one aspect we do not left behind, especially by dealing with an outsourcing environment, is prescriptive solution architecture. A good solution architecture approach ensures agreement about

technical risk mitigation, technological decisions and requirements realization. It also provides team scalability and helps realistic scope definition.

**Agile software construction:** our Improvement Engineering process is the agile portion of our method. It was designed with some agile principles [19] background. Furthermore, as we mentioned before, we adopt some Scrum practices like task board, daily meetings and short sprint-like iterations. Nevertheless, an important question arises: it is well known that refactored architecture and exploratory requirements are some of the central aspects on most of agile methods, but we used opposite approaches in order to ensure all advantages we just mentioned above. So how can one improve software factory dynamics by an agile approach if requirements management and architecture are prescriptive? The exploratory nature and dynamics of agile development is possible to be exploited because of the Implementation Scenario approach presented below.

**Implementation scenario:** we introduce the concept of implementation scenario: a particular risk-based subset of requirement – even non-functional ones – conveniently chosen by the Solution Architect depending on the lifecycle moment, with focus on implementation for architectural prioritizing. On RUP, the requirement is the basic unit of architectural prioritizing, which implies on waste of time and risk management power by implementing non-architectural portions of code on elaboration phase. The main implication of implementation scenario is to split requirements and prioritize only those portions appropriated to each development phase. Besides, implementation scenarios play an important role in prescriptive and agile integration: the architecture is prescriptive, but the architectural stabilization and software construction are made by exploratory identification and prioritization of scenarios.

## VI. OUTSOURCING PROPOSAL

The outsourcing contribution of this work is related with its built-in structure, which consists of a well-defined customer-provider interface from three points of view: (1) activity assignment, (2) artifact custody and (3) the method design itself.

**Activity assignment:** specifies either the activity is performed on customer side, provider side or mutually. We represented it graphically by different background colors for each activity inside the Activity Diagrams.

**Artifact custody:** we defined a supplementary mapping of artifacts to be developed and delivery along the development lifecycle. For each project to be executed, this mapping must be instantiated. The artifacts are classified by (1) phase (initial phase where it is produced), (2) required or not, (3) intervention (how the customer works on the artifact: revision, tracking, approval or formal validation) and (4) format/tool (what tool or file format is used to produce the artifact).

**Method design:** although the method defines formal requirements and prescriptive architecture, the software

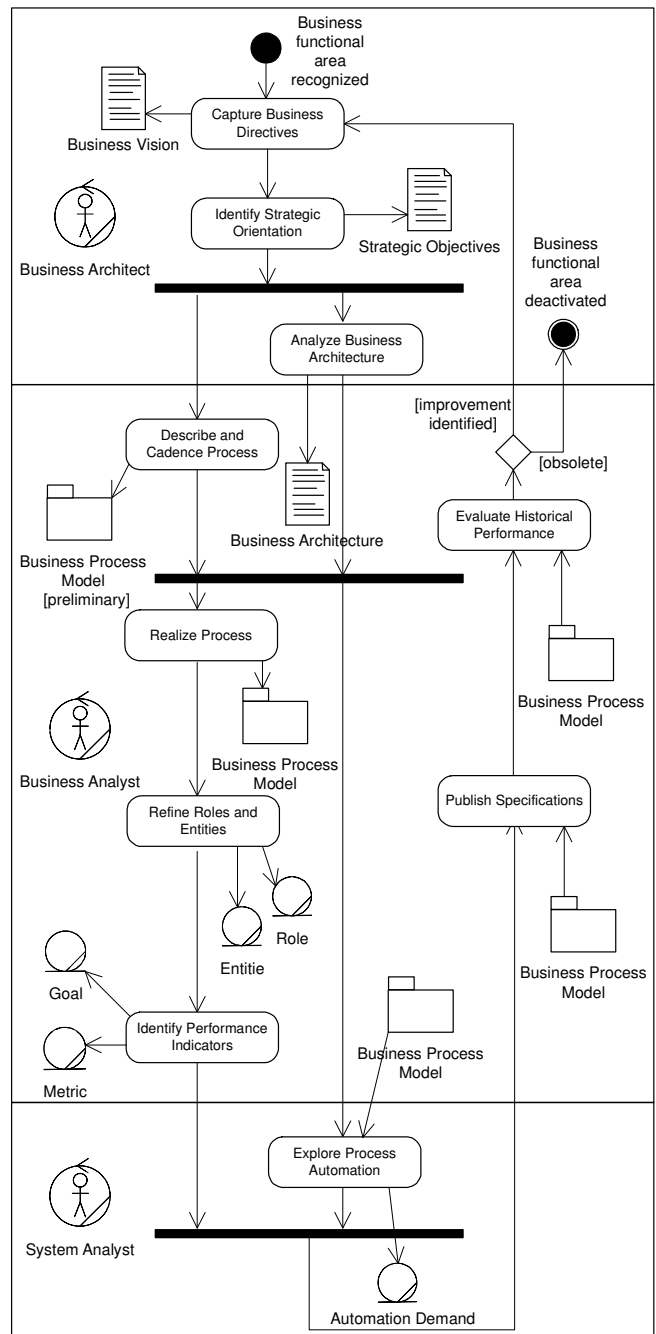


Fig. 1 – The “Process Engineering” process.

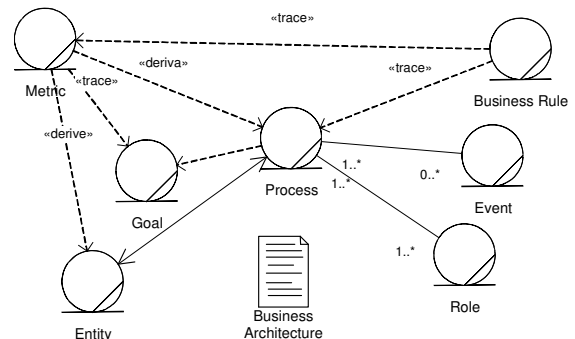


Fig. 2 – Process Engineering related entities and relationships

construction (inside the software factory) is based on exploratory implementation scenarios in an agile fashion. The main technical outsourcing interface point is the architecture: it is being prescriptively designed with mutual participation, but the Solution Architect from provider side has freedom to exploratory choose implementation scenarios while inside the Improvement Engineering process. The main management outsourcing interface point is the formal requirements management, which give the basis for scope, budget, schedule, quality management and nonetheless acceptance of the project's outcome.

## VII. EMPIRICAL TESTS AND RESULTS

As mentioned before, an experienced IT company has been testing this method for more than six months on four small-size and medium-size projects (ranging from 1000 to 7000 hours each), playing the "provider role". The "customer role" is being played by one of its clients (a big enterprise group that has an IT department).

The Project Manager and Technical Director were interviewed in order to give feedback about the improvements they could perceive from the new method. The main improvements reported are listed below:

**Risk management:** the use of implementation scenario greatly improved technical risk management and therefore the architecture stabilization. By taking exploratory risk-driven part of requirements one can focus on really important aspects to architecture stabilization and, thus, improving its effectiveness and mitigating later phases risk of delay.

**Productivity:** the use of agile approach to software construction improved the software factory dynamics and flexibility. The increase of efficiency on development was specially perceived by the use of shorter iterations and physical task board.

## VIII. CONCLUSIONS

We presented a semi-prescriptive software development lifecycle method by defining seven IT processes. The method was built on top of a well-defined Process Engineering foundation as a positive attitude against the methodological neglecting often present on software methods creation [1].

A summary of main contributions is shown below:

- A solution of the agile-disciplined integration was proposed. Essentially, formal requirement management and prescriptive architecture were taken from the disciplined world and exploratory (risk-driven) implementation scenarios and agile practices for software factory were taken from agile world. In other words, this way we could improve development dynamics without harming scoping and scalability.
- A built-in method support to software development outsourcing was proposed. This support consists of a process-centered interface between customer and provider based on activity assignment, artifact custody and the overall method design itself.

- Informal tests have been carried out and significant results were reported, mainly concerning risk management and productivity.

One can also verify that all issues pointed out on Section I were addressed by this approach.

## IX. FUTURE WORK

Additions to this work are already being carried or planned. We are going to define process metrics for this method in order to obtain tangible quantitative evidence for validation. Other future directions are those about integration with other IT areas (like business consulting) and continual refinement.

This method is also intended to be applied in combination with Application Lifecycle Management (ALM) as part of a broader IT governance model to multi-sourcing environments.

## X. REFERENCES

- [1] Ramsin R. and PAIGE, R. F., "Process-Centered Review of Object Oriented Software Development Methodologies", *ACM Computing Surveys*, Vol. 40, No. 1, Article 3, February 2008.
- [2] Boehm, B. and Turner, R., "Balancing Agility and Discipline: A Guide for the Perplexed", Addison Wesley, Reading, MA, 2003.
- [3] Abrahamsson, P., Warsta, J., Siponen, M. T., and Ronkainen, J., "New directions on agile methods: A comparative analysis", *Proceedings of the International Conference on Software Engineering (ICSE)*, 2003.
- [4] Boehm, B. and Turner, R., "Management challenges to implementing agile processes in traditional development organizations", *IEEE Software*, 22, 5 (September/October), 30–39, 2005.
- [5] Nerur, S., Mahapatra, R., and Mangalaraj, G., "Challenges of migrating to agile methodologies", *Communication of the ACM* 48, 5 (May), 73–78, 2005.
- [6] Kruchten, P., "Rational Unified Process: An Introduction", 3rd ed, Addison-Wesley, Reading, MA, 2003.
- [7] Kroll, P. and Kruchten, P., "The Rational Unified Process Made Easy: A Practitioner's Guide to Rational Unified Process", Addison-Wesley, Reading, MA, 2003.
- [8] Jacobson, I., Booch, G., and Rumbaugh, "Unified Software Development Process", Addison-Wesley, Reading, MA, 1999.
- [9] Unified Modeling Language Specifications (v2.0), Object Management Group, 2004 - <http://www.omg.org/technology/documents/formal/uml.htm>
- [10] Ambler, S.W., Nalbone, J., and Vizard, M. J., "The Enterprise Unified Process: Extending the Rational Unified Process", Prentice-Hall, Englewood Cliffs, NJ, 2005.
- [11] Schwaber, K. and Beedle, M., "Agile Software Development with Scrum", Prentice-Hall, Englewood Cliffs, NJ, 2001.
- [12] Schwaber, K., "SCRUM development process", *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'95)*, 1995.
- [13] Gonzalez, R., Gascoa, J. and Llopis, J., "Information systems outsourcing: A literature analysis", Volume 43, Issue 7, October 2006.
- [14] Lee, J., Huynh, M. Q., Kwok, R. C. and Pi, S., "IT outsourcing evolution---: past, present, and future", *Communications of the ACM*, Volume 46, Issue 5, May 2003.
- [15] Pei, Z., Zhen-xiang, Z. and Chun-ping, H., "Study on Critical Success Factors for IT Outsourcing Lifecycle", *Int Conf. on Wireless Communications, Networking and Mobile Computing*, 2007.
- [16] Martin, A., Biddle, R. and Noble, J., "When XP Met Outsourcing", *Extreme Programming and Agile Processes in Software Engineering, Lecture Notes in Computer Science*, Volume 3092, 2004.
- [17] Yalaho, A., "A Conceptual Model of ICT-Supported Unified Process of International Outsourcing of Software Production", *IEEE International Enterprise Distributed Object Computing Conference Workshops*, IEEE Computer Society, 2006.
- [18] ITIL - Information Technology Infrastructure Library. <http://www.itil-officialsite.com>
- [19] Bech, K., et al. 2001. Manifesto for agile software development - <http://agilemanifesto.org>.

# A MODEL DRIVEN METHOD FOR DATA WAREHOUSE

Leopoldo Zepeda<sup>1</sup>, Elizabeth Ceceña<sup>1</sup>, Jorge Rivas<sup>1</sup>, Javier cano<sup>1</sup>, Nelly Condory<sup>2</sup>, Matilde Celma<sup>2</sup>  
Tecnológico de Culiacán, , Sinaloa, Mexico  
Universidad Politecnica de Valencia, Valencia, España  
lzepeda@itc.edu.mx

Keywords: Data Warehouse, Model Driven Architecture, OLAP model.

Abstract: Nowadays, it is getting more common to develop Data Warehouse (DW) systems. To deal with the construction of this kind of data bases current DW methods should provide the mechanisms that facility the integration of user requirements in addition to the availability structure of the operational database. This paper presents a Model Driven method that achieves the integration of user requirements with the multidimensional structures available in the operational data sources.

## 1 INTRODUCTION

A DW is a database used for analytical processing whose principal objective is to maintain and analyze historical data. As yet there is not a well-defined strategy for the design and construction of this kind of systems; there are different ways in which a DW system can be built. For instance, some approaches design the DW starting from a detailed analysis of the operational data sources; others start from determining the information requirements of DW users. A more complex way could be support integration of user requirements and the multidimensional structures available in the operational database. Our proposal introduces some contributions in this perspective because we think that defining the multidimensional schema is one of the most critical steps in the overall DW development process which demand knowledge of operational data sources and user requirements. In this context, it is necessary to provide a methodological guide that helps DW developers in the construction process of this kind of systems. This is achieved by developing a method for the integration of user requirements with the multidimensional structures available in the operational data base. We think that this process should be tackled following a Model Driven approach. This paper is structured as follows: in section 2 we review previous approaches on DW

design. Section 3 introduces our method. Finally, Section 4 draws some conclusions and future works.

## 2 RELATED WORKS

Model Driven Architecture (MDA) is a standard that addresses the cycle of designing, deploying, and managing applications by using models in software development [1]. MDA separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform. Thus, MDA encourages specifying a Platform Independent Model (PIM) which contains no information specific to the platform. Then, this PIM can be transformed into a Platform Specific Model (PSM) in order to include information about the specific technology. MDA also presents a Computation Independent Model (CIM), this model describes the system within its environment and shows what the system is expected to do. Using a series of transformations, also called model transformations, the software system is developed from a PIM to source code.

In [2] a method for developing multidimensional schemas is presented. The design method starts from an existing Entity Relationship (ER) schema, derives a multidimensional schema, and provides implementations in terms of relational tables as well

as multidimensional arrays. In [3] the authors present a DW design method. The design of a conceptual schema is carried out by producing a fact schema for each fact, which can be derived from an ER schema using an algorithmic procedure. The above contributions are concerned with DW conceptual design starting only from conceptual operational schemas. The most valuable contribution of those proposals is that they incorporate concepts and notations to the model to reflect graphically multidimensional aspects. Nevertheless, the use of proprietary notation is a deficiency, since turns these methods in particular and isolates solutions. On the other hand, to the best of our knowledge, only one effort has been development for aligning the design of DWs with the general MDA paradigm. In [4] the authors apply MDA to the logical stage of the DW development. They present a set of transformation rules between the OLAP and Relational PIM. However, the OLAP PIM is very simple and does not offer the necessary details used in real models.

### 3 OUR METHOD

Our approach aims to perform an automatic analysis of the operational data sources in order to discover the implicit multidimensional (OLAP) schemas in it. After that, we reconcile these schemas with end-user requirements. This process is divided into four steps. First one identifies end-user requirements. An automatic process on the other hand (step 2), starts with the identification of the multidimensional elements in the operational database and creates different models of the relational PIM. In step 3, a set transformation rules produce the OLAP schema from the relational models. Finally, an integration process and a model to text transformation process generate the SQL code for the creation of the OLAP schema. In the next sections, we briefly describe each step. An example is presented in several parts according to the section been discussed.

#### 3.1 Defining the CIM

According to MDA, a CIM must describe the requirements of the system. We specify the early requirements of a DW by means of a goal model. We propose two steps to define the goal model:

1. **Goal identification.** We identify the set of goals that both, the system together and an actor must achieve to accomplish each requirement. The set of identified goals are organized in a Goal Refinement Tree (GRT). The GRT represents the

goals that the actor can achieve when interacting with the DW.

*Part I: Example.-* In this section we provide an example of our approach, related to the information system of a self-service store. In our example, two main domain stakeholders are identified: *sales manager* and *offer manager*. The strategic goals of the *sales manager* are: G1.- *Increase return on investment* and G2.- *Increase customer fidelity*. For instance the strategic goal *Increase return on investment* may be AND decomposed into G.1.1.- *Increase sales volume* and G1.2.- *Increase sales profit*. Likewise, *increase sales volume* might be OR decomposed into G.1.1.1.- *Increase consumer appeal* or G.1.1.2.- *Expand market*. In our example, at least two well-established tasks can be to *Increase sales profit*: G.1.2.1.- *Increase sales price* or G.1.2.2.- *Lower production costs*. The partial representation of this model is shown in figure 1.

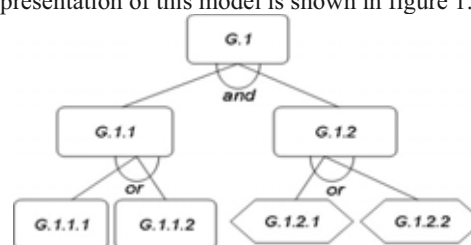


Figure 1: Partial goal model

2. **Goal Description.** To accomplish the goal defined by each leaf task included in the GRT, we describe the set of actions to obtain some goal of the organization. This description is completed by using UML Activity Diagrams. During this step, each task of the GRT is related to the actions that stakeholders consider necessary in order to satisfy each task. In these diagrams, we show the actions performed to obtain some task, indicating the roles that are in charge of each activity, and the data required and produced by each activity. Data appear as objects that flow between activities. We refer to these objects as Data Objects (DO). We distinguish two different types of DOs. 1) Output DO: the system provides actors with information about data. 2) Input DO: the system is waiting for the user to introduce some data.

*Part II: Example.-* Figure 2, shows an activity diagram for the description of the task G.1.2.1: *increase sales price* task. This task is related with two actions: *analyze the margin profit* and the *quantity sold*. The activity diagram starts with the selection of an individual action. Thus, for instance if the selected action is *quantity sold*, this action will search information that matches with the

information provided by the DW user through an Input DO (*Year, Promotion and Store*). In order to make goal descriptions, we propose the definition of an information template (see Table I) for each task identified. In each template we describe the information in detail by means of a list of properties associated to the task.

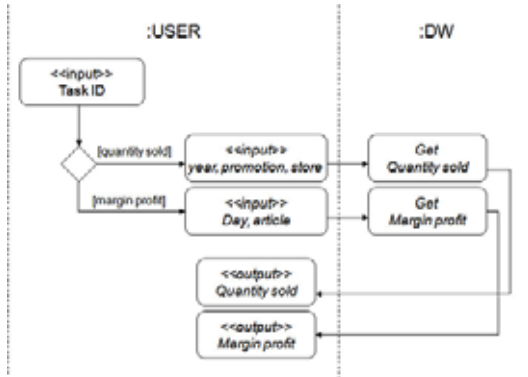


Figure 2: Task description

According to the information showed in table I, the information that the DW must store about the *increase sales price* task is: *Promotion, Year, Day, Store, Quantity sold* and *Margin profit*.

Table I: Information template.

Name	Data type	DO
Promotion	String	Input
Year	Date	Input
Store	String	Input
Day	String	Input
Quantity sold	Number	Output
Margin profit	Number	Output

The information template can be interpreted to select a candidate multidimensional schema, the items listed in the DOs section are considered as measures and dimensions in the multidimensional schema. Then, the Input DO defines the variables that may cause changes to measures (dimension) and each Output DO contributes to a measure. The information template of Table I can be interpreted as follows: the Input DO (*Promotion, Year, Day, Store*) detail the dimensions, while the Output DO (*Quantity sold, Margin profit*) details the measures.

### 3.2 Building models of the Relational PIM

The aim of this step is to automatically create different instances of the Relational PIM. For this; we have developed an algorithm that identified the multidimensional elements in the relational schema of the operational database and creates different instances of this PIM. In this section we first

describe the Relational PIM, next we introduce the most relevant steps of the algorithm, followed by a brief explication of each one.

#### 3.2.1. Relational PIM

The CWM relational [5] PIM (figure 3) is structured into a schema class that owns all elements of a relational model. In the relational PIM, a *Table* is used to store *Columns*. Each *table* can contain a Primary Key or multiple Foreign Keys.

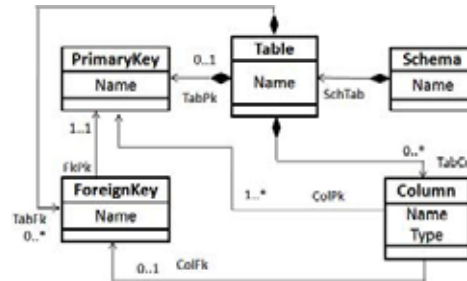


Figure 3: Relational metamodel

#### 3.2.2. The Search Algorithm

The algorithm to get a set of relational models starting from the logical schema of the operational database, consists in performs an exhaustive analysis to it. The goal is identifying the tables that are candidates to be cubes in the OLAP model. Once the tables are identified a search for dimensions and levels must be done. The goal is to add dimensions so we can produce a relational model for each cube identified. The algorithm follows the next three principal steps.

- [S1]- *Identifying cubes*.- A table  $T$  is mapped to a cube  $C$  in the relational model if  $T$  has the following features: *big size cardinality and the possible presence of measures*.
- [S2]- *Identifying measures*.- Each numeric attribute from  $C$  is mapped to a measure  $M$  in the relational model.
- [S3]- *Identifying dimensions and levels*.- Dimensions and levels are identified as follow:
  - a.- Let  $FK$  be a foreign key between the tables ( $C, E$ ), where  $FK$  has multiplicity  $C(1,1), E(0,N)$  and  $C$  is a cube, then:
    - $FK$  is mapped to a dimension  $D$  in the relational model.
    - $E$  is mapped to a level  $L$  of the dimension  $D$  in the relational model.
  - b.- Let  $FK$  be a foreign key between the tables ( $E_j, E_k$ ), where  $E_j$  has been mapped to a level of the dimension  $D$  then:
    - $E_k$  is mapped to a level  $L_k$  of the dimension  $D$  in the relational model.

Part III: Example.- As the main need of the company identified in section 3.1 was to study and analyze the *Sales* process. We focus on part of the operational schema that supports the *Sales* business process (figure 4), where primary keys are underlined and foreign keys dashedlined.

Promotion (Prom\_id, Description)  
 Client (Client\_id, Name)  
 Store (Store\_id, Name)  
 Article (Article\_id, Manufacturer\_id, Price, Description)  
 Manufacturer (Manufacturer\_id, Description)  
 Forecast (Forecast\_id, Article\_id, Store\_id, Total)  
 Ticket (Ticket\_id, Store\_id, Client\_id, Total)  
 Line (Line\_id, Ticket\_id, Prom\_id, Article\_id, Price, Quantity)

Figure 4. The Sales operational database schema

First one the algorithm looks for tables with numeric attributes and big size cardinality (step S1). Following this condition, the set of cubes identified in the operational schema are: *Line* and *Forecast*. Then, the algorithm selects a table from this set, for example: *Line*. Then the numeric attributes *Price* and *Quantity* belonging to the *Line* table are considered measures of the OLAP model (Step S2). According to the step S3, the foreign keys attributes of *Line* (*Ticket\_id*, *Prom\_id* and *Article\_id*) are considered dimensions. A search in the relational schema is done for each table related with *Line* (step S3). Those tables will be considered dimension's levels of the OLAP model associated to *Line*. The set of levels identified are: *Ticket*, *Promotion* and *Article*. The algorithm continues the search of levels following the foreign key chain until the chain end.

### 3.3. Transformation rules

In this section, we describe the OLAP PIM and some of the transformations rules.

#### 3.3.1 OLAP PIM

In the OLAP PIM [5], each *Dimension* is a collection of *Members* (figure 5). *Cubes* are used to store *Measures* and they are related to the *Dimensions* through the *CubeDimensionAssociation* class. *Dimensions* can contain multiple and diverse hierarchical arrangements of *Members* including a specialized *Hierarchies* that support ordering *Members* by *Hierarchy Levels* (*HierarchyLevelAssociation*).

#### 3.3.2. Transformation rules

In this section we describe some of the transformation rules using the diagrammatic notation of the declarative approach of QVT. Each transformation contains the following elements:

- Domains: identifies a set of elements to match in the target model by means of patterns.
- A relation domain: it specifies the kind of relation between domains, since it can be marked as checkonly (C) or as enforced (E).
- When clause: it specifies the pre-conditions that must be satisfied to carry out the transformation.
- Where clause: it specifies the post-condition that must be satisfied by all model elements participating in the relationship.

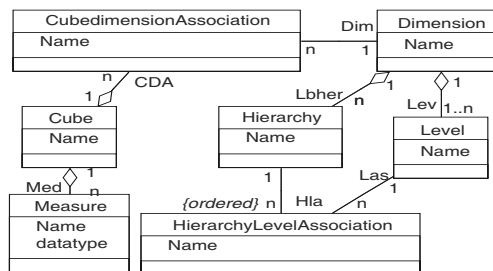


Figure 5. OLAP metamodel

**TableToCube.** According to this transformation rule (figure 6), a candidate cube table gets transformed to a corresponding *Cube*, having the same name of the table. Once this transformation is done, the transformation rules *ForeignKeyToCDA* and *AttributeToMeasure* must be done.

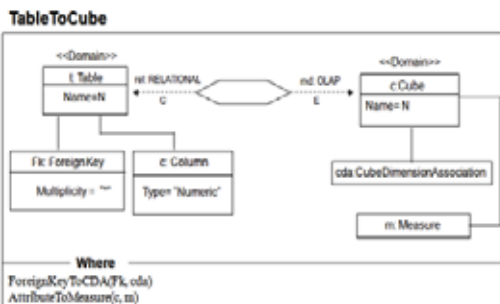


Figure 6. Transformation rule TabletoCube

**ForeignKeyToDim.** In this rule (figure 7), a Foreign Key gets converted to a corresponding *Dimension*, having the same name as the Foreign Key, but prefixed with a "D". The check domain (C arrow) determines the transformation in the following way: 1) each path identified is matching with a *Hierarchy* through the transformation rule *HierarchyAuxToHierarchy*. 2) The goal of the function *Get\_Hierarchies* is identify each path starting from the relationship. 3) Each table in the path is matching with a level through the transformation rule *TableToLevel*. 4) The goal of the function *Get\_levels* is identify each table in the path.



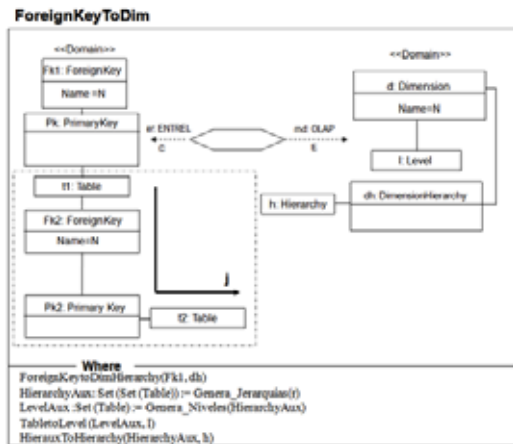


Figure 7 Transformation rule ForeignKeyToDim

*Part IV: Example.-* Using a textual representation a partial transformation for the candidate cube table *Line* can be tracked from T1 to T7. In T1, the candidate cube table *Line* is transformed into a Cube class. Once this transformation is executed, the following transformation rules *AttributetoMeasure* and *ForeignKeyToDimension* are executed (T2-T7).

```
T1:Table (Name="Line") → TableToCube → Cube
(name="Line").
T2:Attribute (Name="Quantity") →
AttributetoMeasure → Measure ("MQuantity")
T3:Attribute (Name="Price") →
AttributetoMeasure → Measure ("MPrice")
T4:ForeignKey (Name="Lin_Tic") →
ForeignKeyToDimension →
Dimension(Name="DLin_Tic")
T5: ForeignKey (Name="Lin-Pro") →
ForeignKeyToDimension →
Dimension(Name="DLin-Pro")
T6: ForeignKey (Name="Lin-Art") →
ForeignKeyToDimension →
Dimension (Name="DLin-Art")
T7:Table (Name="Ticket") → TableToLevel →
Level (Name="LTicket")
```

The candidate multidimensional schema *Line* obtained from the relational schema is shown in figure 8.

### 3.3. Integration and code generation

In this section, we present the rules that analyze the elements produced from user requirements to select and refine the OLAP models and the model to text transformation rules.

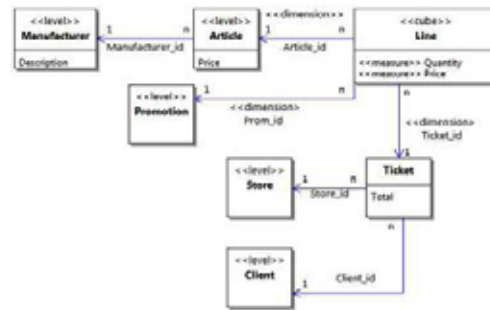


Figure 8. Line model

#### 3.3.1. Schema Matching

On one hand, the algorithm discovers a set of OLAP models. These OLAP models capture the available structures in the operational data base. At the requirements level we specify the data requirements, that is, the detailed information that the DW must recognize in order to properly support tasks that users must perform. Thus, analyzing the information templates we can select the OLAP model that best fit the user requirements. The metrics to which OLAP model acquires them are:

- Corresponding attributes. We must identify attributes from the cubes of each OLAP model that has a correspondence with the measures identified from user requirements.
- Corresponding dimensions. We must identify dimensions from each OLAP model that has a correspondence with dimensions obtained from user requirements.

Table II resume the number of properties of each OLAP model (for the *Analyze ppromotions* task). Based on this information we can select the OLAP model *Line* over the *Forecast* model, because it captures better the user requirements and is supported by the operational database.

Table II: Schema Matching.

Property/Schema	Line	Forecast
Corresponding attributes	1	0
Corresponding dimensions	2	1

Once the model has been selected, it must be manually modified. During this process we can eliminate unnecessary levels, add measures, etc. To better understand this step, we describe the process that eliminates a dimension level.

*Part V: Example.- Eliminate a dimension level.-* Probably, not all of the levels represented in the selected OLAP schema are interesting for DW. Thus, the level must be eliminated. For instance, one may want to classify the information in *Line* cube



# Analyzing the software development process with SyQL and Lagrein

Mirco Bianco  
Center for Applied Software  
Engineering  
Free University of Bolzano  
Via della Mostra, 4  
I-39100 Bolzano-Bozen  
[Mirco.Bianco@unibz.it](mailto:Mirco.Bianco@unibz.it)

Alberto Sillitti  
Center for Applied Software  
Engineering  
Free University of Bolzano  
Via della Mostra, 4  
I-39100 Bolzano-Bozen  
[Alberto.Sillitti@unibz.it](mailto:Alberto.Sillitti@unibz.it)

Giancarlo Succi  
Center for Applied Software  
Engineering  
Free University of Bolzano  
Via della Mostra, 4  
I-39100 Bolzano-Bozen  
[Giancarlo.Succi@unibz.it](mailto:Giancarlo.Succi@unibz.it)

## Abstract

*Mining information from software products metrics and software process data is very hard[14]. Automatic collected data from the source code metrics extractors and from the software development process probes have different formats, it makes difficult to use both at the same time. In this paper, we present a data manipulation language called System Query Language (SyQL), which overcomes problems of other similar languages and allows the user to access data stored in a relational-temporal database. Developers and managers can look at effort data and code metrics by writing very concise SQL-like queries and by using linguistic variables that are unavailable in other existing similar query languages. SyQL helps the user to access temporal data of the software process, providing a set of temporal constructs. Examples of problems solved using SyQL queries and Lagrein (a tool for source code analysis) are provided, evidencing the advantage of the proposed approach.*

**Keywords** Query languages, data warehouses, software metrics, effort, development process.

## 1. Introduction

Mining information from software process and products metrics at the same time is challenging [5]. The relations between them can be different depending on the analysis to perform.

Typically, researchers mine information from relational data warehouses in asynchronous way, using SQL to perform data extraction and other data manipulation tools (Weka, RapidMiner, Matlab, etc) to perform elaboration, such as filtering, clustering, etc. These data warehouses grow up to 1.5 GB/day [14], therefore the asynchronous approach is very time consuming. Moreover, the structure of the data warehouse is usually fairly complex [13].

To overcome such problems we propose a new language: System Query Language (SyQL). SyQL is a

domain specific language based on fuzzy-temporal logic to query a data-warehouse of software process data [15] through “SQL like” queries. SyQL is used inside Lagrein [7] for retrieving and visualizing the historical data about the software development process (code metrics, effort, bugs, etc.). Software development takes place over time. To allow the user to consider the time aspect when evaluating software metrics, SyQL offers the possibility to filter data using temporal conditions.

The paper is organized as follows: section 2 defines the goals of this work, section 3 discusses the related work, section 4 presents our solution, section 5 gives an overview of our automatic metrics collection system, section 6 introduces the syntax of SyQL, section 7 describes how the SyQL query engine works, section 8 shows examples of visualization, finally section 9 draws the conclusion and presents future directions.

## 2. The Goals

SyQL has been designed to achieve the following goals:

- Build an abstraction layer between the user and the tables of the data-warehouse;
- Make the query preparation process against the metrics data warehouse [14] trivial;
- Help software engineers to evaluate software along the timeline;
- Support the evaluation of the effort spent by the developers along the temporal line;
- Help the user to evaluate product quality using simple logic constructs;
- Make the language extensible.

Summarizing, SyQL has a high aggregation capacity and it supports extensible fuzzy logic and temporal functions. The Fuzzy logic is useful for performing qualitative analysis on large datasets, which sometimes is more useful than quantitative analysis, because the user cannot a priori estimate the value of software metrics [19]. The user can miss some important results if

he/she uses a wrong threshold value. Therefore, a fuzzy set encapsulates the “experience” to evaluate a particular metric. By temporal functions, we mean language clauses that help the user to write shorter queries. Temporal functions are needed because the software development process evolves over time, and so queries must include temporal conditions [1][16][18][10].

### 3. Related work

There are several works on languages that can be used to query repositories of software data. The features that appear most relevant to consider are: the capabilities to perform temporal queries on product and process metrics, the possibility to help the user to filter the results through linguistic variables [17] (such as high, medium, low), and the possibility to be used into a general context. In addition it is important to consider some other technical aspects such as: support to combined analysis (software metrics/effort), temporal management, fuzzy logic support, supported programming languages (languages from which the tool is able to extract information for analysis tasks), and object orientation.

In Table 1 we use such criteria to compare some of the most relevant existing work and SyQL.

Language Integrated Query – LINQ [9] is designed to be embedded into another programming language. Therefore, queries can be performed with the same expressive power from a program written either in C# 3.5, VB 9.0, or another .NET language. FuzzySQL [4] is a commercial relational database front-end; it supports fuzzy conditions and it is designed to assist the user during the analysis tasks. .QL [11] is a commercial tool designed to perform code analysis tasks as reverse engineering and discovery of bad code smells. DmFSQL [2] is a general-purpose fuzzy query language data-mining oriented implemented as an Oracle database front-end. SCQL [6] is a domain-specific temporal query language used to retrieve information from a relational database containing information gathered from a source control system. NDepend is an application, which uses CQL (Code Query Language), for extracting information from .NET projects. With this program is possible to extract a lot of information from the source code.

### 4. Our proposal

To enable the final user to perform fuzzy-temporal query against a metrics data warehouse [15] we decided to implement a new query language, SyQL. The reasons of this choice are now discussed, showing the

main differences with those of languages introduced above.

The syntax of SyQL is similar to the one of LINQ [9], but it is designed to achieve different purposes. LINQ is more general and can perform queries on different data sources, while SyQL is tight to a specific data source (the metrics data warehouse [15]). Both of them are fully object oriented; SyQL allows the use of Fuzzy equal operator and temporal tokens, LINQ does not.

The main difference between SyQL and FuzzySQL [4] is that FuzzySQL is a general-purpose relational database front-end, while SyQL is a specific tool to perform information retrieval tasks on metrics data warehouse [14] with additional features to handle temporal analysis of the software development process.

SyQL can be used to perform software metrics and effort analysis, on the contrary .QL [11] can handle only software data. SyQL can perform tasks on different project written in different programming languages, while .QL can perform analysis only on Java projects. SyQL supports the fuzzy logic conditions, .QL does not.

SyQL is completely different from dmFSQL [2], the only evident similarity between them is the fuzzy logic support, because the purposes of these two languages are different.

Both SCQL [6] and SyQL have keywords to manage temporal data. The main difference is that SyQL is designed to be extended to handle different aspects of development process (effort, software metrics, requirements, etc.), while SCQL is designed only to perform information retrieval tasks on software repository data. SCQL has not fuzzy logic support.

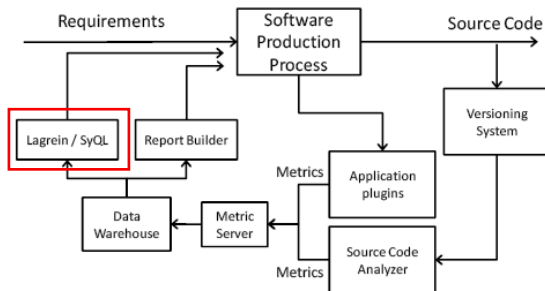
NDepend and SyQL have been designed for achieving different goals. With NDepend is easier keep under control a set of .NET projects, because it is highly integrated with the .NET environment, on the other hand SyQL is more platform independent (it supports also C/C++ and Java) and it wants to help the users to control different aspect of the software development process. With SyQL is possible to visualize and compare the values of a specific metric into a specified time interval (e.g. show the total number of line of code in the last 6 months), with NDepend is possible only to compare two different versions of the code showing the changes. SyQL makes possible running real effort analyses on source code (e.g. compute the total effort spent by the developers on a specific package/namespace), it enables the user to track the bug fixing process showing which methods had been modified during a specific fixing task, with NDepend it is not possible.

**Table 1: Comparison between different query languages.**

Languages	Support to combined analysis (software metrics/effort)	Temporal management	Fuzzy Logic	General Purpose language	Supported programming languages (for analysis task)	Object Orientation
LINQ [9]	NO	NO	NO	YES	None	YES
FuzzySQL [4]	NO	NO	YES	YES	None	NO
.QL [11]	NO	NO	NO	NO	Java	YES
dmFSQL [2]	NO	NO	YES	YES	None	NO
SCQL [6]	NO	YES	NO	NO	None	NO
NDepend <sup>1</sup>	NO	NO	NO	NO	All .NET languages	YES
SyQL	YES	YES	YES	NO	C/C++, Java, C#, VB.NET	YES

## 5. Architecture description

Before presenting the architecture of SyQL and how the results are displayed, we are going to give a brief introduction to our distributed non-intrusive system for collecting software metrics [14]. Figure 1 shows the role of SyQL and Lagrein in the system. The metric collection system is distributed: the applications plugins are installed on the clients and they are able to trace the user activities inside the most common IDEs (Microsoft Visual Studio, Eclipse, etc.); the Source Code Analysis components runs on a standalone machine that takes daily snapshots of the source code from the Versioning System. These components send the collected data to the Metric Server using Apache XML-RPC protocol implementation. Then, the Metrics Server organizes these data and stores them inside the relational data warehouse. The extracted information are delivered to the managers and to the developers in two possible ways, either by an automatic statically generated report (using Eclipse BIRT) or by Lagrein/SyQL in a "dynamic/visual" way.



**Figure 1: The System Architecture.**

## 6. Language description

We introduce the structure of the language through an example.

```

[01] FROM Class c, Method m
[02] WHERE c.getFullName() =
[03]        m.getDefClassFullName()
[04] AND c.getEffort(YESTERDAY) IS High
[05] SELECT c.getFullName(),
[06]        c.getEffort(TODAY - 1 'day'),
[07]        COUNT(m)
[08] GROUP BY c.getFullName(),
[09]          c.getEffort(TODAY - 1 'day');
  
```

The above query returns a collection of class names, the related effort spent by the developers since yesterday, and the number of methods for each class. The first row introduces the *FromClause*, which could contain one or more *FromElement*(s). Each of them is composed by two literals, the former identifies the concept type, the latter declares the concept name (like in SQL). The second, third and fourth rows introduce the *WhereClause*. In the example there are two conditions: an equal join condition and a fuzzy condition. The fuzzy condition evaluates the effort spent yesterday by the developers. The method *c.getEffort(...)* is a Java method that returns a value. In the fifth, sixth, and seventh rows the *SelectClause* is shown. This is a non empty collection of *MethodCall*(s) and/or aggregation functions (like Count, Sum, Max, Min, etc.). In the last two rows we declare the *GroupByClause*, which is similar to SQL one. As happens in others similar query languages [9] [11], we decide to put the *FromClause* at the beginning of the query for allowing to use the auto completion in *Where*, *Select*, and *GroupBy* clauses.

<sup>1</sup> <http://www.ndepend.com/>

## 7. How the query engine works

### 7.1 Concepts and methods

The extensibility is one of the main requisite of SyQL engine, different concepts (the non-terminal symbol *FromElement*) and methods (the non-terminal symbol *MethodCall*) used into a SyQL query are shipped in a separate library. This allows us to implement new concepts and new methods during the entire lifecycle of SyQL. Another advantage is that SyQL acts as an abstraction layer between the user and the data-warehouse. Therefore, we can modify the schema of the data-warehouse without affecting the user, if the library is updated properly.

Implementing a new concept in SyQL has only one requirement: an instance of one concept must be an entry of a relation defined with a SQL statement. In this way, we can perform the mapping between the SyQL concepts and the tables. The materialization of the object is performed through a constructor, which takes as input an entry of the relation defined above.

All the methods of a concept class that can appear in the *SyQLExpression* are annotated in two different ways. An annotated method can become part of an external or an internal calculable condition. A method can be annotated as external if the returned value is present in one column of the defining concept relation, otherwise it must be annotated as internal. If a condition, which is represented by an instance of *SyQLRelationalExpression*, is composed by at least one internal calculable method, it must be evaluated into the SyQL query engine, otherwise it can be evaluated by the

query engine of the underlying DBMS. The *FuzzyExpression(s)* are internal by default.

### 7.2 Query Execution

The SyQL query engine works on top of the DBMS (Figure 2).

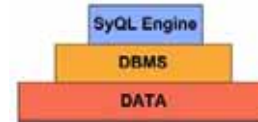


Figure 2: The Data Layers.

The SyQL query engine has been implemented without the need of developing a sophisticated query planner and executor. The idea is to push as much conditions as possible into the query engine of the underlying DBMS, in this way we obtain better time performance because the SyQL query engine does not execute any join. To perform it correctly, we convert the conditions that appear in the *WhereClause* into an equivalent Conjunctive Normal Form (CNF) formula using Boolean algebra and the De Morgan's theorem. The CNF notation is very helpful, because a block of OR conditions can be processed by the underlying DBMS query engine only if all the conditions (inside the block) are evaluated as external, otherwise the block of conditions must be evaluated by the SyQL query engine. A condition is evaluated as an external one if all the predicates (of the condition) are external, otherwise a condition is evaluated internally. The query execution workflow is shown in Figure 3.

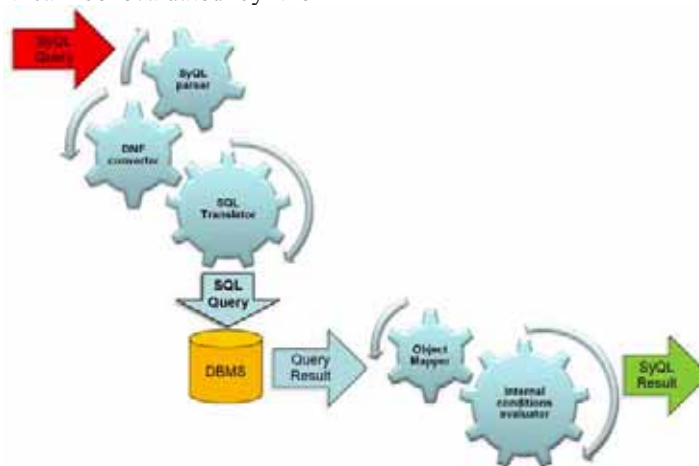


Figure 3: The SyQL Query Workflow.

To perform always this conversion, we convert the parsed formula into an equivalent Disjunctive Normal Form (DNF) formula. Then, we convert it into an

equivalent CNF formula doing the Cartesian product among all the condition contained into the AND blocks. The most critical component for the perform-

ance is the internal condition evaluators, usually internal conditions require a lot of computation, because most of them need to fetch data from the database. To address this problem we adopted two solutions: 1) sorting these conditions according to their cost, the cost is estimated by the developer of the SyQL libraries during the implementation; 2) evaluating these conditions in parallel taking advantage of the modern parallel/multicore hardware architectures.

## 8. Query visualization

SyQL query results may produce a large quantity of data. Extracting useful information from a large temporal series may be difficult for a human user. Inspect a large software system (about 1,000 classes) on a temporal line of one month (20 working days) generates about 20,000 values per selected class metric, assuming that we collect one metrics snapshot per day without specify any filtering condition. If we perform queries on methods instead on classes, the reader can easily understand how the number of results grows up. Computer animation can easily be a useful and intuitive solution for displaying evolving datasets [12]. We solve this problem mapping the query results inside the metric views of Lagrein. Mapping these results it is straight forward because the SyQL query engine is written in Java, the common implementation technology simplifies the integration between the two tools.

### 8.1 Introducing query visualization by examples

#### Example 1:

In this example we visualize the growth of the classes (in term of LOC) where the developers have spent high effort during the last four days.

```
FROM Class c, Chron chr
WHERE chr.getDate() >= TODAY - 4 'days'
      AND chr.getDate() < TODAY
      AND c.getEffort(TODAY - 4 'days', TODAY)
           IS High
SELECT c.getLOC(chr.getDate());
```

The result of this query can be visualized either in an Evolution matrix (Figure 4) or in a Evolution Chart. The query above is a collection of *ClassLOC* instances. The *ClassLOC* class implements the interface *ClassMetric*. Through this interface is possible to retrieve the date, the class owner, and the value of the metric. In this way, it is possible to create an animated view of the growth of the classes in the last four days.

It is also possible repeat this query for all the software metrics collected by the source code analyzer (Cyclo-matic Complexity, Halstead Volume, CK metrics [3]).

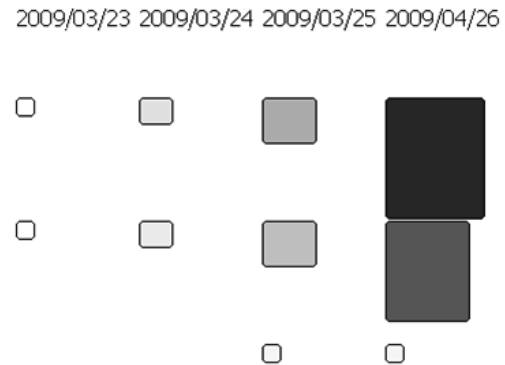


Figure 4: Evolution Matrix

#### Example 2:

It is also possible to create static views of the system. In this example we perform selection of the classes with high value of Coupling Between Objects (CBO).

```
FROM Class c
WHERE c.getCBO(TODAY) IS High
SELECT c;
```

The result of this query (static result) can be visualized in several views (Figure 5) available in Lagrein (e.g., Inheritance tree, Dependency graph, etc).

## 9. Conclusion and future work

This paper discussed a possible approach for visualizing and mining software metrics and software process data. The whole architecture of the metric collection system and the language structure of SyQL have been presented and a comparison to existing systems is provided, showing that SyQL can go further than the other existing languages. The query execution workflow has been discussed. As a proof of concept a set of examples has been provided to the reader. Now we are using this language to build training dataset for estimating the fault-proneness of a method. We will embed these models into SyQL concept libraries, so we will enable the language user to estimate the fault-proneness of a method simply from a SyQL queries.

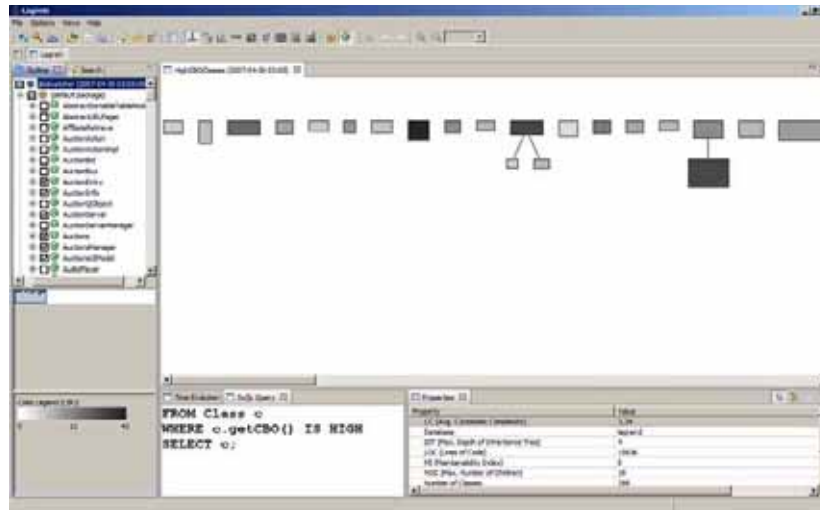


Figure 5: Inheritance Tree of High CBO Classes

## References

- [1] M. Böhlen, J. Gamper, and C. Jensen. Multi-dimensional aggregation for temporal data. In *Advances in Database Technology - EDBT 2006*, pages 257-275, 2006.
- [2] R. Carrasco, M. Vila, and F. Araque. dmFSQL: A language for data mining. In *Proceedings of the 17th International Conference on Database and Expert Systems Applications*, pages 440-444, 2006.
- [3] S. Chidamber and C. Kemerer. A metrics suite for object oriented design. *IEEE TSE*, 20(6):476-493, 1994.
- [4] E. Cox. FuzzySQL a tool for finding the truth: the power of approximate database queries. *PC AI*, 14(1):48-51, 2000.
- [5] F. Fioravanti, P. Nesi. Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems. *IEEE TSE*, 27(12):1062-1084, 2001.
- [6] A. Hindle and D. M. German. SCQL: a formal model and a query language for source control repositories. In *Proceedings of the 2005 workshop on Mining software repositories*, pages 1-5, 2005.
- [7] A. Jermakovics, R. Moser, A. Sillitti, and G. Succi. Visualizing software evolution with lagrein. In *OOPSLA Companion*, pages 749-750, 2008.
- [8] M. Karaila and T. Systa. Applying template meta-programming techniques for a domain-specific visual language – An industrial experience report. In *Proceedings of the 29th international Conference on Software Engineering*, pages 571-580, 2007.
- [9] E. Meijer, B. Beckman, and G. Bierman. LINQ: reconciling object, relations and XML in the .NET framework. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 706-706, 2006.
- [10] B. Moon and F.V. Lopez. Efficient algorithms for large-scale temporal aggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):744-759, 2003.
- [11] O. d. Moor, M. Verbaere, E. Hajiyev, P. Avgustinov, T. Ekman, N. Ongkingco, D. Sereni, and J. Tibble. Keynote Address: .QL for source code analysis. In *Proceedings of the Seventh IEEE international Working Conference on Source Code Analysis and Manipulation*, pages 3-16, 2007.
- [12] M. Pinzger, H. Gall, M. Fischer, and M. Lanza. Visualizing multiple evolution metrics. In *Proceedings of the 2005 ACM Symposium on Software Visualization*, pages 67-75, 2005.
- [13] K. Ramamurthy, A. Sen, and A.P. Sinha. Data Warehousing Infusion and Organizational Effectiveness. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 38(4):976-994, 2008.
- [14] M. Scotto, A. Sillitti, G. Succi, and T. Vernazza. A non-invasive approach to product metrics collection. *Journal of System Architecture*, 52(11):668-675, 2006.
- [15] M. Scotto, A. Sillitti, G. Succi, and T. Vernazza. Non-invasive collection of software metrics: some issues and experiences. In *Sharing experiences on agile methodologies in open source software development*, Polimetrica Publisher, Italy, pages 31-38, 2006.
- [16] J. Yang and J. Widom. Incremental computation and maintenance of temporal aggregates. *The VLDB Journal*, 12(3):262-283, 2003.
- [17] L. A. Zadeh. The Concept of a Linguistic Variable and its Application to Approximate Reasoning. *Information Science*, 8:199-249, 1975.
- [18] D. Zhang, A. Markowetz, V. Tsotras, D. Gunopulos, and B. Seeger. Efficient computation of temporal aggregates with range predicates. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 237-245, 2001.
- [19] S. Zhang, J. Lu, and C. Zhang. A fuzzy logic based method to acquire user threshold of minimum-support for mining association rules. *Information Sciences*, 164(1-4): 1-16, 2004.



# Performance Analysis of a Deductive Database with a Semantic Web Reasoning Engine: ConceptBase and Racer

Simone A. Ludwig, Craig Thompson, Kristofor Amundson  
Department of Computer Science, University of Saskatchewan, Canada  
ludwig@cs.usask.ca

## Abstract

*Knowledge engineering is a discipline concerned with constructing and maintaining knowledge bases to store knowledge of various domains and using the knowledge by automated reasoning techniques to solve problems in domains that ordinarily require human logical reasoning. Therefore, the two key issues in knowledge engineering are how to construct and maintain knowledge bases, and how to reason out new knowledge from known knowledge effectively and efficiently. The objective of this paper is the evaluation of a Deductive Database system with a Semantic Web reasoning engine. For each system a knowledge base is implemented in such a way that comparable performance measurements can be performed. The performance and scalability are evaluated for class and instance queries.*

## 1. Introduction

Knowledge engineering is a discipline concerned with constructing and maintaining knowledge bases to store knowledge of the real world in various domains and using the knowledge by automated reasoning techniques to solve problems in domains that ordinarily require human logical reasoning. Therefore, the two key issues in knowledge engineering are how to construct and maintain knowledge bases, and how to reason out new knowledge from known knowledge effectively and efficiently.

Knowledge-based systems (KBS) use human knowledge to solve problems which normally requires human intelligence. A KBS shell is a software environment containing a knowledge acquisition system, the knowledge base itself, inference engine, explanation subsystem and user interface. The core components are the knowledge base (human knowledge represented by e.g. IF-THEN rules) and the inference engine (forward or backward chaining).

MYCIN [1] is an example of a rule-based expert system which was designed for the diagnosis of

infectious blood diseases. MYCIN has been developed without using a modeling framework, opposed to a few frameworks which were developed to help during the knowledge engineering process such as CLIPS (C Language Integrated Production System) [2] or JESS (Java Expert Systems Shell) [3]. CLIPS is a productive development and delivery expert system tool which provides a complete environment for the construction of rule and/or object based expert systems. JESS is a rule engine and scripting environment written in Java. With JESS, one can build software that has the capacity to "reason" using knowledge supplied in the form of declarative rules. JESS uses an enhanced version of the Rete algorithm [4] to process rules which is a very efficient mechanism for solving the difficult many-to-many matching problem. CommonKADS [5] is known for having a structure of the Expertise Model and Model-based and Incremental Knowledge Engineering (MIKE) [6], which relies on formal and executable specification of the expertise model as the result of the knowledge acquisition phase.

Another approach for reasoning is Deductive Databases, where data is described by logical formulas, usually in a restricted subset of first-order logic. These formulas are intended to specify part of the external world relevant to the application at hand, called the application world. Thus, a Deductive Database is a logical representation of the application world. Therefore, the semantics of Deductive Databases are based on mathematical logic. A user queries a Deductive Database by submitting a goal. Goals are also logical formulas. A correct answer to a goal provides values for the variables of the goal that make this query logically follow from the database. Hence, the semantics of query answering in Deductive Databases is based on the notion of logical consequences developed in mathematical logic. Besides formulas specifying the database and queries, a Deductive Database can also contain integrity constraints: logical conditions which the database must satisfy at any given moment [7].

The latest reasoning technology for the Web is the Semantic Web, which vision is to make the Web

machine-readable, allowing computers to integrate information and services from diverse sources to achieve the goals of end users. It allows to reason about the content when Web pages and services are augmented with descriptions of their content. Semantic Web technologies are used in many ways to transform the functionality of the Web by enriching metadata for Web content to improve search and management; enriching descriptions of Web services to improve discovery and composition; providing common access wrappers for information systems to make integration of heterogeneous systems easier; and exchanging semantically rich information between software agents. Ontology languages [8] were created to augment data with metadata. The most recent ontology for the Web is called OWL (Web Ontology Language). OWL builds on a rich technical tradition of both formal research and practical implementation.

This research was motivated by the fact that reasoning on the Web becomes ever more important due to the advancement of Web services and service computing on the whole. However, not much research has been conducted into the evaluation of the performance and scalability of reasoning on the Web. Furthermore, no comparison between an established reasoning tool, namely the deductive database, has been done. The objective of this paper is the evaluation of a Deductive Database system with a Semantic Web reasoning engine. For each system a knowledge base is implemented in such a way that comparable performance measurements can be performed.

The paper is outlined as follows. In Section 2, both systems, ConceptBase and Racer are described. In Section 3, the knowledge base, queries, measurement methodology and setup are outlined. Section 4 presents the performance analysis of both systems exploring the load time and the scalability of classes and instances. The findings and conclusions are given in Section 5.

## 2. Description of Both Systems

ConceptBase was chosen as the Deductive Database system to compare with the Semantic Web reasoning engine Racer. The two systems are described in more details in the subsections below.

### 2.1. ConceptBase

ConceptBase has been used in a number of applications at various universities in Europe. The ConceptBase system, developed since 1987, seeks to combine deductive rules with a semantic data model based on Telos [9] (described further below). The

system also provides support for integrity constraints [10]. ConceptBase is free software available for download, and the user interface is java based. Furthermore, ConceptBase uses the client-server architecture, and has a fairly extensive Application Programming Interface (API) for writing clients in Java, C or C++.

ConceptBase is a deductive object-oriented database management program intended for conceptual modeling. It uses O-Telos which is a version of the logical knowledge representation language Telos, which includes deductive and object-oriented features. O-Telos is based on Datalog, which is a subset of Prolog.

ConceptBase allows for logical, graphical and frame views of databases. The ConceptBase graph editor allows one to visualize the relationships in the database, as well as adding and modifying the classes, individuals, and relationships. Queries are represented as classes that have membership constraints. Within the database, all classes, instances, attributes, rules and constraints are represented as objects that may be updated at any time. However, there is not an option to cascade changes, so it is easy to add information at any time, but it can be difficult to remove information.

### 2.2. Semantic Web Technologies: Protégé and Racer

The Semantic Web technology used to create an ontology to represent the application domain was Protégé [11], a Java-based, free ontology editor developed by Stanford Medical Informatics at the Stanford University School of Medicine. It provides a knowledge base that allows the user to create formal rules for a knowledge representation system to reason through. After developing a taxonomy and creating rules the ontology can be exported in OWL format, which is similar to XML in syntax and includes the descriptions of the classes and individuals along with their explicit relationships. Protégé also provides a Java API that allows OWL files to be imported and represented as Java classes. The API has the capability to connect to a knowledge representation system, such as RACER (Renamed ABox and Concept Expression Reasoner) [12], allowing implicit relationships to be found.

RACER is commercial software developed by RACER Systems and was used for this research investigation. This software is capable of reasoning through Description Logic TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, tree-conjunctive query answering using an XQuery-

like syntax), such as the ones that are created using Protégé and exported in the OWL format.

### 3. Evaluation

In order to perform a comparison analysis of Racer and ConceptBase, a knowledge base was implemented in both systems. Queries were chosen which return the same results to evaluate class and instance queries. The measurement methodology and setup are described below.

#### 3.1. Knowledge Base

The knowledge base / ontology used for the evaluation is an extension of the pizza ontology supplied with Protégé.

**Table 1. Ontology description of scaling classes**

Ontology size	Number of classes	File size Racer (in KB)	File size ConceptBase (in KB)
1	263	279	27
2	495	565	54
3	727	869	82
4	959	1189	109
5	1191	1536	137
6	1423	1897	163
7	1655	2280	191
8	1887	2683	219
9	2119	3105	246
10	2351	3553	273

The ontology contains classes describing pizzas and ingredients, as well as sandwiches and salads. The dishes (pizzas, sandwiches, salads) were defined in terms of the ingredients they contain. All subclasses in the ontology were given instances, and in some cases higher level classes had instances, so there are nearly as many instances as classes. Some dishes were defined to describe specific foods, such as a BLT (Bacon Lettuce Tomato) sandwich, other dishes such as *vegetarianPizza* were defined to be any pizza without meat or fish. The classes describing specific foods were given necessary conditions, for example, this pizza must have mozzarella as a topping. The other classes, such as *vegetarianPizza*, were given necessary and sufficient conditions, meaning that any pizza that had no meat or fish would be considered a *vegetarianPizza*. Thus, the classes that met the

necessary and sufficient conditions would be subsumed, creating an inferred hierarchy of classes.

ConceptBase on the other hand, required a slightly different modeling technique. It is not possible to create an inferred class hierarchy, thus, in order to have similar reasoning capabilities to the Protégé ontology, *queryClasses* were used. Query classes have constraints describing which individuals may be members of the query class. Thus, with the vegetarian pizza example, members of the vegetarian pizza query class were defined to be any individual that did not have meat, or fish, as an ingredient.

**Table 2. Ontology description of scaling instances (number of classes fixed to 263)**

Ontology size	Number of instances	File size Racer (in KB)	File size ConceptBase (in KB)
1	217	305	46
2	434	321	65
3	651	348	84
4	868	375	104
5	1085	402	123
6	1302	433	142
7	1519	454	162
8	1736	480	181
9	1953	507	200
10	2170	542	220

As knowledge bases consist of classes and instances, the investigation will only focus on class and instance reasoning. In order to measure how good both systems scale, we expanded the ontologies in two directions; (1) scaling of classes and (2) scaling of instances. Table 1 and 2 show the properties of the different ontology sizes used for this investigation. Table 1 contains 10 different ontology sizes, whereby the number of classes is increasing with the size without containing any instances. For the scaling of instances, the class structure of the size 1 ontology (Table 1) is fixed to 263 classes for all different instance ontology sizes.

#### 3.2. API and Queries

The ConceptBase API provides methods to ‘tell’ files to the ConceptBase server, retrieve a named class or individual, find instances of a class or query a class, retrieve attributes of classes or individuals, retrieve superclasses and subclasses, find the class that an

individual belongs to, and get generalizations and specializations from a class. Additionally, several Boolean operations are provided for testing the relationships between classes, or instances, such as *isSuperclassOf* or *isExplicitInstanceOf*. There appeared to be many methods that returned the same, or very similar results in different formats, such as newline delimited, or comma delimited. The redundant queries in ConceptBase returned the same classes, but in different formats, e.g., subclasses can be returned in ConceptBase code syntax, or as a string with one class per line, or with all classes on one line separated by commas, or as a hashset, depending on what the user want to do with the subclasses. Attributes in ConceptBase are tied directly to the class they represent, so all information about attributes is gained through the appropriate class, or instance. The useful methods for obtaining information from the database can be found in the *ICBclient* and *ITelosObjectSet* classes.

The Protégé API allows the user to find descendant classes, classify the taxonomy, compute the inferred hierarchy, compute the inferred types of all individuals, retrieve ancestor classes, retrieve equivalent classes, retrieve subclasses, find individuals belonging to a class, determine the subsumption relationship between two classes, return the superclass of a class, get sub properties, get inverse properties, return the inferred equivalent classes, get the inferred subclasses, get the inferred superclasses, maximum and minimum cardinalities of properties, determine if subclasses are disjoint, determine if a class has a superclass, return the name of an instance, return the namespace of the ontology, return a list of the possible rdf properties, and return rdf types. Properties in Protégé are independent of classes and instances, and thus may be queried directly. The useful methods for gaining information about the model were spread across several classes in the API, namely, *ProtegeOWLReasoner*, *RDFProperty*, *OWLProperty*, *OWLNamedClass* and *OWLIndividual*. Among these classes, there seemed to be several redundant methods. This is because *OWLProperty* inherits from *RDFProperty* and therefore has all the same methods, plus a few more. *ProtegeOWLReasoner* and *OWLNamedClass* have some methods with the same results, the difference is that *ProtegeOWLReasoner* calls RACER, whereas *OWLNamedClass* uses the results from the last time the reasoner was used.

The main type of reasoning of a knowledge base can be divided into two categories, class and instance reasoning. In order to perform a fair analysis of these systems, equivalent queries existent in both systems which perform the same type of reasoning were

chosen: Query 1 and 2 are class queries, and query 3 and 4 are instance queries.

Query 1 returns all subclasses belonging to a particular class: *getDescendentClasses* (Racer); *getAllSubclassesOf*(ConceptBase).

Query 2 returns the superclasses of a particular class: *getSuperClasses* (Racer); *getExplicitSuperClasses* (ConceptBase).

Query 3 returns all individuals that are members of a particular class: *getIndividualsBelongingToClass* (Racer); *getAllInstancesOf*(ConceptBase).

Query 4 returns all classes that an individual or an instance belongs to: *getIndividualTypes* (Racer); *getClassificationOf*(ConceptBase).

### 3.3. Methodology

Bash scripts were used to automate all the measurement runs. The process for each measurement was as follows: start Racer or the ConceptBase server, run the java query, and close Racer or ConceptBase server to clear the cache. This process was repeated 30 times (to guarantee normal distribution) for each query. The Java query file used to perform a query would start by loading the data model into Racer or the ConceptBase server. Then, the java method *System.nanoTime* was used immediately before and after the query, and the difference was calculated to estimate the performance of the query. Each time the java program was executed it would perform only one query, in order to avoid caching issues across queries. *System.nanoTime* was found to give results with a higher precision than *System.currentTimeMillis*, especially as several of the queries took less than one millisecond to execute.

### 3.4. Measurement Setup

The following measurement setup was used for this investigation:

- Hardware configuration (Lenovo M55 with 2.4GHz Intel Core2 CPUs and 2GB of RAM; no hyperthreading).
- Software configuration (Mandriva Linux 2008.1; Java 1.6.0\_03; latest versions of ConceptBase 7.1, Protégé 3.4 and Racer 1.9.2.)

## 4. Results

The evaluation was performed as follows. First, the load times for loading the different ontologies into memory are measured. Afterwards, the scalability of classes and instances are evaluated.

#### 4.1. Load Time of Different Ontology Sizes

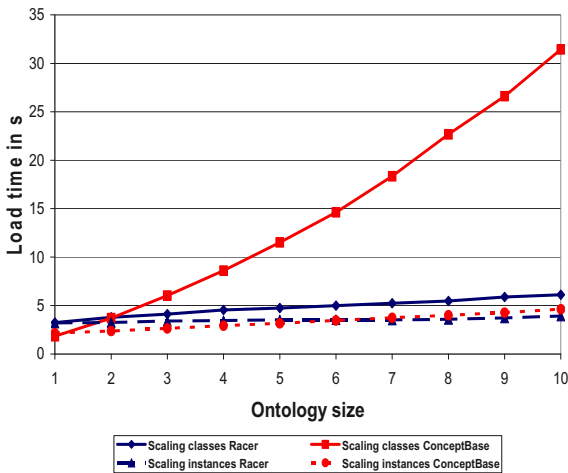


Figure 1. Load time of Racer for scaling of classes and instances

Before queries are run in both Racer and ConceptBase, the knowledge base or ontology needs to be loaded into memory first. Figure 1 shows the load time in seconds for increasing ontology sizes. Two distinctions are made here for either scaling of classes or instances. The scaling of Racer shows a linear distribution with increasing ontology sizes, whereby the scaling of classes has a greater impact on the performance than the scaling of instances. The scaling of classes has a gradient of 0.3, whereas the scaling of instances has a gradient of 0.07. The load time for ConceptBase has a slightly different distribution. The scaling of instances seems to be linear; however, the scaling of classes follows a quadratic distribution. The query times for the scaling of classes are also larger than for the scaling of instances as also observed for Racer. Comparing both systems it can be concluded that the load time of ConceptBase is greater by a factor of 3.01 for the scaling of classes, but is almost similar for the scaling of instances with a factor of 0.95.

#### 4.2. Scaling of Classes

Looking at how the performance scales with increasing ontology sizes (the instance queries would not make sense when querying an ontology without instances), Figure 2 shows the queries run in Racer and ConceptBase (*getSuperClasses*, *getAllSubclassesOf* and *getSuperclassesOf* all have similar query times). It is observed, that both queries, *getDecendentClasses* and *getSuperclasses*, scale in a similar fashion with a quadratic distribution. This is because the same operation is performed, that is the classification of a

new concept. Racer computes more than is required in order to answer these particular class queries. ConceptBase on the other hand shows the measurements of the similar queries with a linear distribution. Instead of both queries scaling in a similar fashion as in Racer, the query time for subclasses is higher than for superclasses. It appears that the performance is dependent on the number of return values. *getAllSubclassesOf* returns 31 to 238 subclasses for ontology size 1 and 10 respectively, while *getSuperclassesOf* returns only 1 superclass for all ontology sizes.

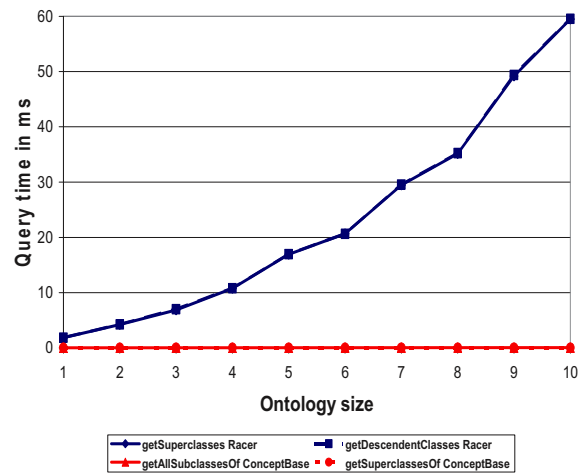


Figure 2. Query time of Racer for scaling of classes

Comparing the class queries executed in Racer and ConceptBase, it shows that ConceptBase scales much better than Racer.

#### 4.3. Scaling of Instances

Figure 3 shows the linear distribution of query times for scaling of instances. For Racer, it shows that the query times for *getIndividualBelongingToClass* are higher than for *getIndividualTypes* queries. *getIndividualTypes* looks at a specific individual and returns all classes of which it is an instance of, whereas *getIndividualsBelongingToClass* has to consider all individuals. The implementation of the operations seem to be quite different and therefore the result can be seen in the query times of both queries in Racer. The instance query, *getAllInstancesOf*, performed in ConceptBase shows a quadratic distribution, whereas the *getClassificationOf* query shows a linear gradient. *getAllInstancesOf* takes longer as the return values range between 47 to 256 for ontology size 1 to 10 respectively, whereas *getClassificationOf* returns always only one return value. The direct comparison of

the query times regarding the scaling of instances of both systems shows that ConceptBase performs better than Racer.

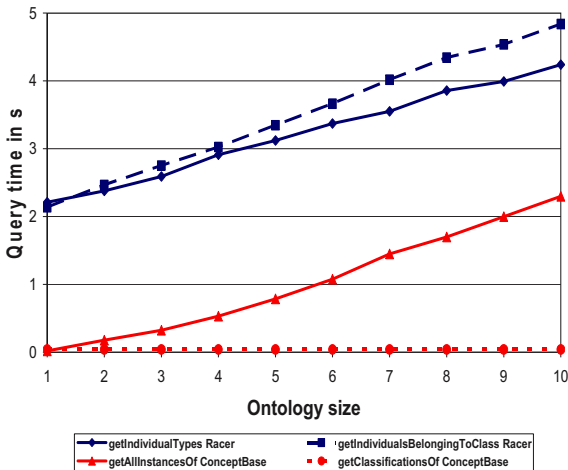


Figure 3. Query time of Racer for scaling of instances

## 5. Conclusion

This paper evaluated a Deductive Database system and a Semantic Web reasoning engine - ConceptBase and Racer. For each system a knowledge base was implemented in such a way that comparable performance measurements could be performed.

The findings revealed that the reasoning capabilities in Racer are richer. Furthermore, queries in ConceptBase run much faster than in Racer. The factors were 61 and 7 for ontology sizes 1 and 10 respectively for ontologies with instances. On the other hand however, the load time to load the different ontologies was better in Racer by a factor of 3.01 for the scaling of classes, but was almost similar for the scaling of instances for which the factor measured was 0.95. The load time for Racer is linear, whereas the load time for ConceptBase seems to have a quadratic growth function. The scaling of classes revealed that class queries take much longer in Racer than in ConceptBases as in Racer all consistencies are being checked before a class query is being performed, measured by a factor of 470. The growth function for Racer for the class queries is quadratic, whereas the growth function for ConceptBase is linear. The scaling of instance queries showed a better performance for ConceptBase than for Racer by a factor of 4.8. However, it seems that ConceptBase is more affected by string processing of the return values of the queries. This means that if a large amount of result values are

returned, the performance decreases in ConceptBase, whereas Racer is not affected by this.

Considering that ontologies are developed incrementally, adding a relatively small increment to a large ontology has a great effect for the loading of this ontology into memory for ConceptBase, whereas the class and instance queries in Racer will have a greater performance reduction than ConceptBase.

As reasoning on the Web has seen a steady increase in the past several years, this evaluation shows that Web reasoning has to speed and scale up with technologies existing for many decades such as deductive databases.

## 6. References

- [1] Shortliffe, E. H., "MYCIN: Computer-Based Medical Consultations", Elsevier Press, New York, 1976.
- [2] CLIPS Website, Last retrieved April 2009 from <http://www.ghg.net/clips/CLIPS.html>.
- [3] JESS Website, Last retrieved April 2009 from <http://herzberg.ca.sandia.gov/jess/>.
- [4] Forgy, C. L., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem". *Artificial Intelligence*, 19(1982) 17-37.
- [5] Schreiber, A. T., Wielinga, B.J., de Hoog, R., Akkermans, H., and van de Velde, W., "CommonKADS: A Comprehensive Methodology for KBS Development", *IEEE Expert*, December 1994, 28-37.
- [6] Angele, J., Fensel, D., and Studer, R., "Developing Knowledge-Based Systems with MIKE", *Journal of Automated Software Engineering*, Volume 5, Number 4, pp. 389-418(30), 1998.
- [7] Voronkov, A., Chapter 1, Lecture notes on Deductive Databases, 2002. [http://www.voronkov.com/dresden/2002/chapter\\_1.ps](http://www.voronkov.com/dresden/2002/chapter_1.ps).
- [8] Gruber, T. R., "Toward principles for the design of ontologies used for knowledge sharing". *Journal of Human-Computer Studies*, Volume 43, Issue 5-6 Nov./Dec. 1995, Pages: 907-928, 1993.
- [9] Jeusfeld, M. and Jarke, M., "From relational to object-oriented integrity simplification", *Proc. Of Deductive and Object-Oriented Databases* 91. Springer-Verlag, 1991.
- [10] Jeusfeld, M. and Staudt, M., "Query optimization in deductive object bases". In G. Vossen J.C. Freytag and D. Maier, editors, *Query Processing for Advanced Database Applications*. Morgan-Kaufmann, 1993.
- [11] Protégé Website, Last retrieved April 2009 from <http://protégé.stanford.edu>.
- [12] Racer Website, Last retrieved April 2009 from <http://www.racer-systems.com>.

# Object Specification Language for Graph Based Conceptual level Multidimensional Data Model

Anirban Sarkar<sup>1</sup>, Sankhayan Choudhury<sup>2</sup>, Nabendu Chaki<sup>2</sup>, Swapan Bhattacharya<sup>1</sup>

<sup>1</sup>National Institute of Technology, Durgapur, West Bengal, India 713209

<sup>2</sup>Department of Computer Science, University of Calcutta, Kolkata, India

[anirban.sarkar@nitdgp.ac.in](mailto:anirban.sarkar@nitdgp.ac.in), [sankhayan@gmail.com](mailto:sankhayan@gmail.com), [nabendu@ieee.org](mailto:nabendu@ieee.org), [swapan.bhattacharya@nitdgp.ac.in](mailto:swapan.bhattacharya@nitdgp.ac.in)

## Abstract

Data Warehouse (DW) design demands a proper methodological support for the designer of DW to specify the system at logical level efficiently from its conceptual level design. This paper proposes an object specification language for Graph Object Oriented Multidimensional Data Model (GOOMD model) for the purpose. The object specification language (GOSL), proposed in this paper permits conceptual multidimensional schemas to be specified in terms of classes and class hierarchies. Further, GOSL provides the mechanism for mapping the On Line Analytical Processing (OLAP) operators as formally defined in GOOMD model, into a set of functions which will operate over the instances of classes and its hierarchies produced by GOSL. The proposed specification language is also useful towards automatic generation of logical model of DW from the conceptual model and its graphical notations.

**Keywords:** *Multidimensional Data Model, OLAP, Object Orientation, Graph Data Model, Logical.*

## 1. Introduction

Data Warehouse (DW) and On Line Analytical Processing (OLAP) in conjunction with multidimensional database are typically used for complex, online and multidimensional analysis of data. In DW design framework Conceptual models with graphical notations are closer to the way users perceive an application domain and the logical models concentrate more to the way designer perceive an application domain.

Designing DW is highly complex engineering task which demands a proper methodological support for the designer to specify the DW system at logical level efficiently from its conceptual level design. But there still is a semantic gap between advanced multidimensional conceptual data models and implementations of such data model at logical level [10]. So, more research scope is there to identify the methodology to preserve all information captured by advanced conceptual multidimensional models in logical design.

Several proposed formal multidimensional data models at conceptual level [1, 2, 3] are compatible to the relational model at logical design phase. But the relational model, however, have serious deficiencies in many aspects [7]. In some other approaches [4, 5, 6] the object

oriented paradigm is considered for conceptual level design of DW. Further, the object oriented specification for multidimensional database has been addressed only in [8] based on GOLD model [6]. But the GOLD model itself lacks from semantic enriched graphical notations and OLAP operational model.

However, the Graph Object Oriented Multidimensional Data Model (GOOMD) [9] provides a novel graph based semantic and simple but powerful algebra to conceptualize the multidimensional data visualization and operational model for OLAP, based on object oriented paradigm. The model provides a set of constructs along with rich set of graphical notations to facilitate the designers of DW and a set of operators for OLAP.

This paper proposes an object specification language for GOOMD model to facilitate the designer of DW. The language is used to specify the system at logical level as well as to provide the direction towards the mapping of the system from conceptual to logical level [Fig 1]. Using the proposed object specification language, conceptual multidimensional schemas can be specified in terms of classes and class hierarchies. Further, it provides the mechanism for mapping the OLAP operators as formally defined in GOOMD model, into a set of functions which will operate over the instances of classes and its hierarchies produced by the same specification language.

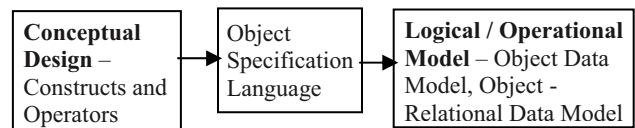
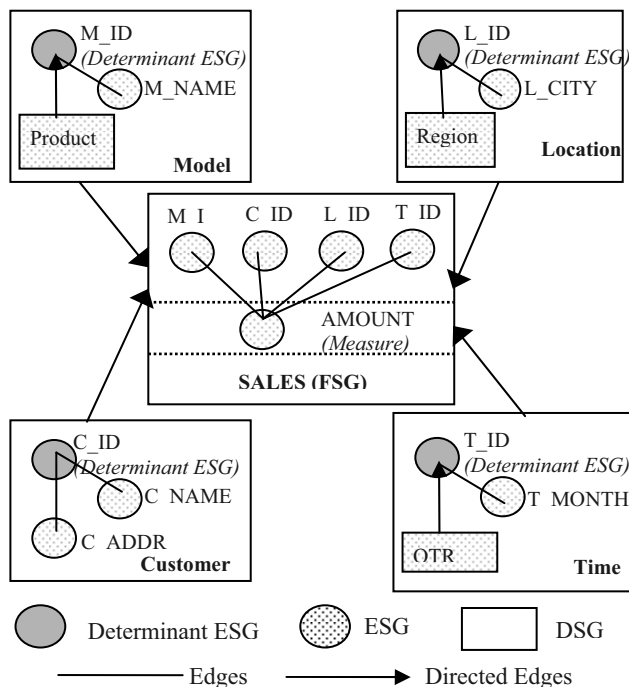


Fig 1: Transformation of Conceptual to Logical Model

## 2. GOOMD Model with Example

In this section we will summarize the basic concepts of GOOMD model [9]. The GOOMD model is the core of the comprehensive object oriented model of a DW containing all the details that are necessary to specify a data cube, the dimensions, the classification hierarchies, the description of fact and measures attributes. It allows the entire multidimensional database to be viewed as a Graph (V, E) in layered organization. At the lowest layer, each vertex represents an occurrence of an attribute or measure, e.g. product name, day, customer city etc. A set of vertices semantically related is grouped together to construct an *Elementary Semantic Group (ESG)*. On next,

several related ESGs are group together to form a Contextual Semantic Group (CSG) – the next upper layer constructs to represent any context of business analysis. A set of vertices of any CSG, those determine the other vertices of the CSG, is called *Determinant Vertices* of said CSG. This layered structure may be further organized by combination of two or more CSGs as well as ESGs to represent next upper level layers From the topmost layer the entire database appears to be a graph with CSGs as vertices and edges between CSGs. *Dimensional Semantic Group (DSG)* is a type of CSG to represent a dimension member, which is an encapsulation of one or more ESGs along with extension and / or composition of one or more constituent DSGs. *Fact Semantic Group (FSG)* is a type of CSG to represent a fact, which is an inheritance of all related DSGs and a set of ESG defined on measures. In order to materialize the *Cube*, one must ascribe values to various measures along all dimensions and can be created from FSG. Two types of edges has been used in GOOMD model, (i) directed edges to represent the one – to – many associations between different CSGs and (ii) undirected edges between constituent ESGs and determinant ESGs to represent the association within the members of any CSG.



**Fig 2: Schema for Sales Application in GOOMD Model**

Let consider an example, based on Sales Application with sales *Amount* as measure and with four dimensions – *Customer*, *Model*, *Time* and *Location* with the set of attributes  $\{C\_ID, C\_NAME, C\_ADDR\}$ ,  $\{M\_ID, M\_NAME, P\_ID, P\_NAME, P\_DESC\}$ ,  $\{T\_ID, T\_MONTH, Q\_ID, Q\_NAME, YEAR\}$  and  $\{L\_ID, L\_CITY, R\_ID, R\_NAME, R\_DESC\}$  respectively. *Model*, *Time* and *Location* dimensions have upper level hierarchies say *Product*, *QTR* and *Region* respectively. Then in the notation of GOOMD model, there will be four

DSGs  $\{D_{Customer}, D_{Model}, D_{Location}, D_{Time}\}$  with hierarchy. The FSG for the database can be described as  $F_{Sales} = \{DET(D_{Customer}), DET(D_{Model}), DET(D_{Location}), DET(D_{Time}), E_{AMOUNT}\}$ , where  $E_{AMOUNT}$  is the ESG defined on the measure *AMOUNT*. The schema from the topmost layer has shown in Fig 2.

GOOMD model also provides algebra of OLAP operators those will operate on different semantic groups. The *dSelect* ( $\pi$ ) operator is an atomic operator and will extract vertices from any CSG depending on some predicate P. The *Retrieve* ( $\sigma$ ) operator extracts vertices from any *Cube* using some constraint over one or more dimensions or measures. The *Retrieve* operator is helpful to realize slice and dice operation of OLAP. The *Aggregation* ( $\alpha$  and  $+\alpha$ ) operators perform aggregation on *Cube* data based on the relational aggregation function like SUM, AVG, MAX etc. on one or more dimensions. *Aggregation* operators are helpful to realize the roll-up and drill down operations of OLAP. GOOMD model also provides the definitions of the operators like Union ( $\cup$ ), Intersection ( $\cap$ ), Difference ( $-$ ), Cartesian Product ( $\times$ ) and Join ( $\Join$ ), which are operated on any CSG or *Cube*.

### 3. Object Specification Language for GOOMD Model (GOSL)

The concept of any multidimensional data model consists of three basic construct namely, (1) Dimensions, where each can consist of a multi-level classification hierarchy, (2) Facts and (3) Measures In object oriented concept different classes need to specify for Dimension members and Fact constructs type. Object identification or OID must address the key attributes specification. In the context of GOOMD model, the construct like DSG and FSG will realize the dimension member class and fact class respectively. The determinant ESG will realize the OID for a specific semantic construct. Further, it is important to note that, the dimension hierarchy level can be represented by corresponding inheritance tree of dimension classes.

Object Specification Language for GOOMD Model or GOSL proposed in this section will be used to specify the classes and its hierarchies corresponding to conceptual level multidimensional schemas defined using GOOMD model. Further GOSL includes the mechanism of mapping the OLAP operators into a set of functions which will operate on the instances of the classes produced by the specification language.

#### 3.1 Class Definitions Using GOSL

In GOSL, the specification of a class is a description of the structure and behavior of the objects belonging to specific semantic group. Each specification of a class begins with the word **class** and consists of a number of sections or paragraphs, according to the formal specification of the GOOMD model. To serve the purpose we have defined two type of Class definition (1) **Regular**



**Class Template** [Fig 3] to specify the CSG of innermost layer which do not have any constituent CSG. Only the DSGs of innermost layer and the DSGs without having any constituent DSG can be represented through regular class definition. **(2) Complex Class Template** [Fig 4] to specify the CSG of upper layers which have atleast one constituent CSG either connected by link of association relation with their parents, for example any DSG or FSG of Top Most Layer.

```

Class <class name>
Type
  DSG
OID
  <visibility> Determinant ESG: type
Fixed Attributes
  <visibility> ESG name: type;
Derived Attributes
  <visibility> Derived ESG name: type;
Methods
  Constructor Functions;
  Destructor Functions;
  Union (List of ESG) return Derived ESG;
  Intersection (List of ESG) return Derived ESG;
  CartesianProduct(List of ESG) return Derived ESG;
End Class

```

**Fig 3: Regular Class Template for GOOMD Model**

```

Class <class name>
Type
  [DSG | FSG]
Parent Class
  Constituent CSGs Name;
OID
  <visibility> Determinant ESG: type
Fixed Attributes
  <visibility> ESG|DSG name: type;
Derived Attributes
  <visibility> ESG name: type;
Measures
  <visibility> ESG on Measure: type;
  Additivity_Constraint : [None / List of DSGs / All]
Methods
  Constructor Functions;
  Destructor Functions;
  Union (List of ESG) return Derived ESG;
  Intersection (List of ESG) return Derived ESG;
  CartesianProduct(List of ESG) return Derived ESG;
End Class

```

**Fig 4: Complex Class Template for GOOMD Model**

For any Dimension class specification, the generalized dimension class name will be specified in *Parent Class* Section. If there does not exist any parent class for some Dimension class, then either the constant literal *No\_Parent* will be used or the section can be omitted from the class definition. *OID* must address the key attributes specification, through which any instance or object of the class can be identified uniquely. By default the OID attributes are inheritable i.e. visibility mode is of protected type. *Fixed Attributes* are the list of static

attributes those are defined for a specific fact or dimension class. For complex class, the attributes can be defined on ESG or constituent DSG connected using association relation with the parent DSG. Since, an attribute can be of complex type i.e. an object of some class. *Derived Attributes* are the list of attributes derived from static attributes by the object specific function defined in *Methods* section. Constructor and Destructor functions will be used to create and to destroy instances or objects of specific Class. The set of functions, those will operate on the attribute type of class templates also will be defined in *Methods* section (Fig 3 and Fig 4). *Measures* section must address the specification of measures attributes and the additivity constraints on measures. The visibility mode of the measure attributes can be considered as protected if there is a possibility of declaration of a derived fact class by reusing the existing fact class.

**Table 1: Representing GOOMD Model Operators into GOSL Function**

Operators of GOOMD Model	Notation	GOSL Function Syntax
dSelect	$\pi$	dSelect(ESG DSG, Predicate) return ESG DSG;
Retrieve	$\sigma$	Retrieve (FSG, List of Derived DSG) return FSG
Aggregation	$\alpha$  $+\alpha$	Aggregation (FSG, List of DSG, Measure ESG, Additivity_Constraint, Aggregation Function Identifier) return FSG;  +Aggregation (FSG, List of DSG, Measure ESG, Constraint, Aggregation Function Identifier) return FSG;
Union, Intersection, Cartesian Product, Difference	$\cup, \cap, \times, -$	Union (List of ESG DSG FSG) return ESG DSG FSG; InterSection(List of ESG DSG FSG) return ESG DSG FSG; CartesianProduct(List of ESG DSG FSG) return ESG DSG FSG; Difference(List of ESG DSG FSG) return ESG DSG FSG;
Join	$ x $	Join(List of FSG   Derived FSG, CON Specification) return FSG;

### 3.2 Function Definitions Using GOSL

The OLAP operators as defined in GOOMD Model can be mapped into functions with atleast one of the arguments as related construct class type i.e. fact class or dimension class, so that the function can manipulate the set of instances of the said class type. These functions need to be mapped out side of any class definitions. The several operators which will operate on attribute type of class templates can be defined as the member function of the class. These operators are UNION, INTERSECTION,

DIFFERENCE and CARTESIAN PRODUCT as defined in GOOMD model [9]. In Table 1 we have summarized the set of GOSL functions mapped from OLAP operators as defined in GOOMD Model.

### 3.3 Example of GOSL

The proposed GOSL is simple, powerful, expressive, and has been drawn from basic concept of object orientation. It can express the concepts, graphical notations and the OLAP operators of conceptual multidimensional data model like GOOMD.

Recalling the *Sales Application* example described in Fig 2 the *Product* DSG is most inner most layer construct and do not have any constituent DSG. Using GOSL, the *Product* DSG can be described using Regular Class template. The upper layer DSG like *Model* is an encapsulation of several ESGs along with an extension of DSG Product. So *Model* DSG can be described using Complex Class template as shown in Fig 5. Further, in the *Sales Application* example the only measure attribute is *Amount*. The *Sales FSG* can be represented using Complex Class Template as shown in Fig 6.

Since any FSG will inherit the Determinant ESGs from the related top most layer DSGs, which in combine will act as OID attributes for the FSG and inherit automatically. Henceforth the type of the OID attributes for the FSG Class template definition is not required to specify explicitly.

```

Class Model
Type
    DSG
Parent Class
    Product;
OID
    Protected MID: Integer
Fixed Attributes
    Private MDES: String;
Methods
    Constructor Functions for Model;
    Destructor Functions for Model;
    ..... Other Methods.....
End Class
    
```

Fig 5: Class Template for Model DSG

### 4. Conclusions

In this paper an object specification language (GOSL) has been proposed for the conceptual level multidimensional data model called GOOMD Model. GOSL can express the concepts, graphical notations and the OLAP operators of the GOOMD model at logical level through a systematic approach. GOSL is used to specify the conceptual multidimensional schemas in terms of classes and class hierarchies and the OLAP operators in terms of GOSL functions. The expressive power of the specification language also has been demonstrated using typical examples.

The proposed object specification language, in general, can be used with any conceptual multidimensional data model with proper mapping scheme. The feature makes it more useful for automatic generation of logical model from the conceptual model and its graphical notations.

```

Class SALES
Type
    FSG
Parent Class
    Model, Customer, Location, Time
OID
    {M_ID, C_ID, L_ID, T_ID}
Measures
    Private Amount: Float;
    Additivity_Constraint : None
Methods
    Constructor Functions for SALES;
    Destructor Functions for SALES;
    ..... Other Methods.....
End Class
    
```

Fig 6: Class Template for Sales FSG

### 5. References

- [1] Nectaria Tryfona, Frank Busborg, Jens G. Borch Christiansen, "starER: a conceptual model for data warehouse design", *Proceedings of the 2<sup>nd</sup> ACM Int. workshop on Data warehousing and OLAP*, Nov 1999.
- [2] Matteo Golfarelli, Dario Maio and Stefano Rizzi, "The Dimensional Fact Model: A Conceptual Model for Data Warehouses", *Int. Journal of Cooperative Information Systems*, Vol 7 (2-3), PP 215-247, 1998.
- [3] Aris Tsois and Nikos Karayannidis and Timos K. Sellis, "MAC: Conceptual data modeling for OLAP", Booktitle: *Design and Management of Data Warehouses*, 2001.
- [4] Sergio Luján-Mora, Juan Trujillo and Il-Yeol Song, "A UML profile for multidimensional modeling in data warehouses", *Data & Knowledge Engineering*, vol 9, Issue 3, PP 725 – 769, December 2006.
- [5] Nicolas Prat, Jacky Akoka and Isabelle Comyn-Wattiau, "A UML-based data warehouse design method", *Decision Support Systems*, Vol 42 (3), PP 1449 – 1473, December 2006.
- [6] J. Trujillo, "The GOLD Model: An Object-Oriented ConceptualModel for the Design of OLAP Applications", *Doctoral Dissertation*, Languages and Information Systems Dept., Alicante University, Spain, June 2001.
- [7] Senko M. E., "Information Systems: Records, relations, set, entities and things", *Information Systems*, Vol. 1.1, PP 3–13, 1975.
- [8] Juan Trujillo, Manuel Palomar, Jaime Gómez, "The GOLD definition language (GDL): an object oriented formal specification language for multidimensional databases", *Proceedings of the ACM symposium on Applied Computing*, Vol 1, March 2000.
- [9] Anirban Sarkar, Swapan Bhattacharya, "The Graph Object Oriented Multidimensional Data Model: A Conceptual Perspective", *16<sup>th</sup> Int. Conference on Software Engineering and Data Engineering (SEDE)*, pp 165 – 170, July 2007.
- [10] Stefano Rizzi, Alberto Abelló, Jens Lechtenbörger, Juan Trujillo, "Research in data warehouse modeling and design: dead or alive?", *Proceedings of the 9<sup>th</sup> ACM Int. Workshop on Data warehousing and OLAP*, November 2006.

# A Framework for Trajectory Data Preprocessing for Data Mining

Luis Otavio Alvares, Gabriel Oliveira, Carlos A. Heuser, Vania Bogorny

Instituto de Informatica – Universidade Federal do Rio Grande do Sul  
Porto Alegre – Brazil

{alvares,gpaoliveira,heuser,vbogorny,}@inf.ufrgs.br

## Abstract

*Trajectory data play a fundamental role to an increasing number of applications, such as traffic control, transportation management, animal migration, and tourism. These data are normally available as sample points. However, for many applications, meaningful patterns cannot be extracted from sample points without considering the background geographic information. In this paper we present a framework to preprocess trajectories for semantic data analysis and data mining. This framework provides two different methods to add semantic geographic information to the important parts of trajectories from an application point of view. It was implemented as an extension of Weka.*

## 1. Introduction

The increasing use of GPS devices to capture the position of moving objects demands tools for the efficient analysis of large amounts of data referenced in space and time. Current analysis over trajectories of moving objects have basically to be performed manually. Another problem is that most techniques for the analysis of this kind of data and more sophisticated approaches as data mining algorithms consider only the raw trajectories, that are generated as pure  $(x,y,t)$  coordinates. In the last years, some works have been developed for trajectory data analysis, such as [5], in particular for discovering dense regions or similar trajectories. However, these approaches consider only the geometric properties of trajectories, what is very limited for many real applications.

GPS and other electronic devices that capture moving object trajectories do not collect the background geographic information. We claim that for several real applications there is a need to preprocess trajectories to add additional information that gives to trajectories more meaningful characteristics. Indeed, this should be the first step, before any trajectory data analysis. We claim that the first additional information to be considered, is the geographic context where trajectories are captured.

Figure 1 shows an example where we can observe the

necessity of extra information to understand trajectories. Figure 1 (left) shows an example of a geometric trajectory, in which the objects move to the same region at a certain time. Considering a pure geometric approach where only the trajectory points themselves are used for mining it could only be discovered that the four trajectories meet in a certain region, or the trajectories are dense in this region at a certain time. In Figure 1 (right), considering the background geographic knowledge, the moving objects go from different hotels (H) to meet the Eiffel Tower at a certain time. From these trajectories with some semantics, the moving pattern *from Hotel to Eiffel Tower* could be discovered. In this example, it is clear that the origin of the trajectories is in *sparse locations* that have the same semantics (it is a hotel). A pure geometric trajectory data mining algorithm would not be able to discover such semantic pattern.

In [2] we presented a *spatial* framework to automatically preprocess geographic data for data mining. In this paper we present an intelligent *spatio-temporal* framework to preprocess trajectories, in order to transform trajectory sample points in a higher level of abstraction, adding geographic semantics to trajectories.

The main contribution of this work is a framework to allow a user to both analyze and mine trajectories in a high level of abstraction, considering the needs of the application. This framework implements two different methods to add semantics to raw trajectories: one is based on the intersection of trajectories with places relevant to the application and the other is based on the speed of the trajectory. Furthermore, different classical data mining algorithms can be applied in the data mining step.

The remaining of this paper is structured as follows: Section 2 introduces some concepts of semantic trajectories, Section 3 presents the proposed framework for trajectory data analysis and mining, Section 4 presents an implementation of the framework and some experiments, and Section 5 concludes the paper and suggests directions of future works.

## 2. Basic Concepts

Recently Spaccapietra [8] introduced the first conceptual model for trajectory data, with two key concepts: stops and

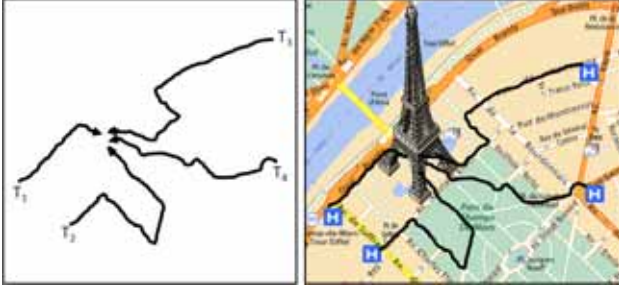


Figure 1. (left) Geometric (raw) trajectories and (right) semantic trajectories

moves. *Stops* are important places of the trajectory from an application point of view, where the moving object has stayed for a minimal amount of time. *Moves* are subtrajectories between two consecutive stops.

To better understand what stops and moves are, we present one formal model where geographic object types are defined a priori by the user as the important places of the trajectory. This model has been introduced in [1] for querying trajectories, but it is not the only way to formally define stops and moves. It will be briefly presented in the following subsections.

### 2.1. Trajectory Samples and Candidate Stops

Trajectory data are normally available as sample points.

**Definition 1** A *sample trajectory* is a list of space-time points  $\langle p_0, p_1, \dots, p_N \rangle$ , where  $p_i = (x_i, y_i, t_i)$  and  $x_i, y_i, t_i \in \mathbf{R}$  for  $i = 0, \dots, N$  and  $t_0 < t_1 < \dots < t_N$ .

To transform trajectory sample points into *stops* and *moves* it is necessary to provide the important places of the trajectory which are relevant for the application. These places correspond to different spatial feature types (spatial object types). For each relevant spatial feature type that is important for the application, a minimal amount of time is necessary, such that a trajectory should continuously intersect this feature in order to be considered a stop. This pair is called candidate stop.

**Definition 2** A *candidate stop*  $C$  is a tuple  $(R_C, \Delta_C)$ , where  $R_C$  is a polygon in  $\mathbf{R}^2$  and  $\Delta_C$  is a strictly positive real number. The set  $R_C$  is called the *geometry* of the candidate stop and  $\Delta_C$  is called its *minimum time duration*.

An *application*  $\mathcal{A}$  is a finite set  $\{C_1 = (R_{C_1}, \Delta_{C_1}), \dots, C_N = (R_{C_N}, \Delta_{C_N})\}$  of candidate stops with mutually non-overlapping geometries  $R_{C_1}, \dots, R_{C_N}$ .

In case that a candidate stop is a point or a polyline, a polygonal buffer is generated around this object, and thus it is represented as a polygon in the application, in order to overcome spatial uncertainty.

### 2.2. Stops and Moves

**Definition 3** Let  $T$  be a trajectory and let

$$\mathcal{A} = \{C_1 = (R_{C_1}, \Delta_{C_1}), \dots, C_N = (R_{C_N}, \Delta_{C_N})\}$$

be an application. Suppose we have a subtrajectory  $\langle (x_i, y_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1}), \dots, (x_{i+\ell}, y_{i+\ell}, t_{i+\ell}) \rangle$  of  $T$ , where there is a  $(R_{C_k}, \Delta_{C_k})$  in  $\mathcal{A}$  such that  $\forall j \in [i, i + \ell] : (x_j, y_j) \in R_{C_k}$  and  $|t_{i+\ell} - t_i| \geq \Delta_{C_k}$ , and this subtrajectory is maximal (with respect to these two conditions), then we define the tuple  $(R_{C_k}, t_i, t_{i+\ell})$  as a *stop of  $T$  with respect to  $\mathcal{A}$* .

A *move of  $T$  with respect to  $\mathcal{A}$*  is one of the following cases: (i) a maximal contiguous subtrajectory of  $T$  in between two temporally consecutive stops of  $T$ ; (ii) a maximal contiguous subtrajectory of  $T$  in between the initial point of  $T$  and the first stop of  $T$ ; (iii) a maximal contiguous subtrajectory of  $T$  in between the last stop of  $T$  and the last point of  $T$ ; (iv) the trajectory  $T$  itself, if  $T$  has no stops.

When a move starts in a stop, it starts in the last point of the subtrajectory that intersects the stop. Analogously, if a move ends in a stop, it ends in the first point of the subtrajectory that intersects the stop.

It is important to notice that the place where a stop occurs is a spatial feature (relevant geographic object) which is intersected by a trajectory for the minimal amount of time. This spatial feature will enrich the trajectory with its spatial and non-spatial information. For instance, if a hotel is a stop, its geometry and the non-spatial attributes (e.g. name, stars, price) is information that can be further used for both querying and mining trajectories.

**Definition 4** A *Semantic Trajectory  $S$*  is a finite sequence  $\{I_1, I_2, \dots, I_n\}$  where  $I_K$  is a stop or a move.

### 3. The proposed framework

Figure 2 shows an interoperable framework with support to the whole discovery process. It is composed of three abstraction levels: data repository, data preparation, and data mining.

At the bottom are the geographic data repositories, stored in GDBMS (geographic database management systems), constructed under OGC [6] specifications. On the top are the data mining toolkits or algorithms. In the center is the trajectory data preparation level which adds semantics to trajectories according to the application domain. In this level the data repositories are accessed through JDBC connections and data are retrieved, preprocessed, and transformed into the format required by the mining tool/algorithm.

There are three main modules to implement the tasks of trajectory data preparation for mining: Clean Trajectories, Add Semantics, and Transformation, which are described in the sequence.

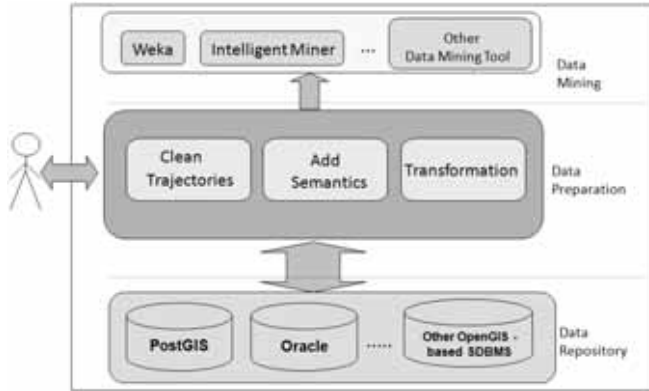


Figure 2. The Semantic Trajectory Mining Framework

### 3.1. Clean trajectories

The Clean Trajectories module performs many verifications over the trajectory dataset in order to eliminate noise, what is very common in this kind of data, and assure that the trajectory dataset is in the format required by the Add Semantics module.

Some of the verifications include: i) the calculated speed between two consecutive points should not be greater than a specified threshold; ii) the trajectory points should be in a temporal order; iii) the trajectories should not have more than one point with the same timestamp; iv) each trajectory should have a given minimum number of points.

### 3.2. Add Semantics

To prepare trajectory data to data mining, the main step is to add semantics to these trajectories. We do that by using the concepts of stops and moves. Two algorithms have been developed so far. The first one, introduced in [1], considers the intersection of a trajectory with the user-specified relevant feature types for a minimal time duration (candidate stops), which we call IB-SMoT (Intersection-Based Stops and Moves of Trajectories).

In general words, the algorithm verifies for each point of a trajectory  $T$  if it intersects the geometry of a candidate stop  $R_C$ . In affirmative case, the algorithm looks if the duration of the intersection is at least equal to a given threshold  $\Delta_C$ . If this is the case, the intersected candidate stop is considered as a stop, and this stop is recorded. If a point does not belong to a subtrajectory that intersects a candidate stop for  $\Delta_C$  it will be part of a move.

Figure 3 illustrates this method. In the example, there are four candidate stops with geometries  $R_{C1}$ ,  $R_{C2}$ ,  $R_{C3}$ , and  $R_{C4}$ . Let us consider a trajectory  $T$  represented by the space-time points sequence  $\langle p_0, \dots, p_{15} \rangle$  and  $t_0, \dots, t_{15}$  are the time points of  $T$ . First,  $T$  is outside any candidate stop, so we start with a move. Then  $T$  enters  $R_{C1}$  at point  $p_3$ . Since the duration of staying inside  $R_{C1}$  is long enough,  $(R_{C1}, t_3, t_5)$  is the first stop of  $T$ , and  $\langle p_0, \dots, p_3 \rangle$

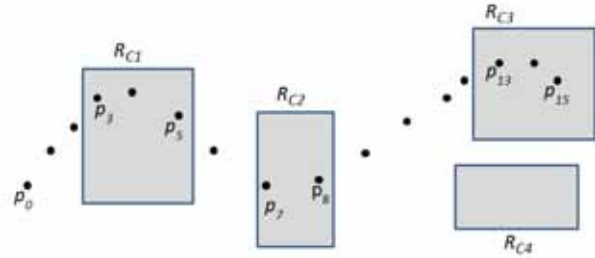


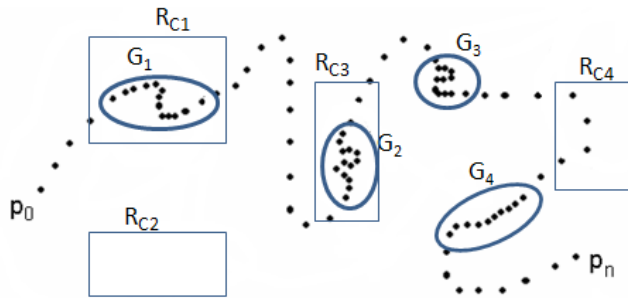
Figure 3. Example of a trajectory with four candidate stops and two stops

is its first move. Next,  $T$  enters  $R_{C2}$ , but for a time interval shorter than  $\Delta_{C2}$ , so this is not a stop. We therefore have a move until  $T$  enters  $R_{C3}$ , which fulfills the requests to be a stop, and so  $(R_{C3}, t_{13}, t_{15})$  is the second stop of  $T$  and  $\langle p_5, \dots, p_{13} \rangle$  is its second move.

The second algorithm is called CB-SMoT [7], and is a clustering method based on the variation of the speed of the trajectory. The intuition of this method is that the parts of a trajectory in which the speed is lower than in other parts of the same trajectory, correspond to interesting places. CB-SMoT is a two-step algorithm. In the first step, the slower parts of one single trajectory are identified, using a spatio-temporal clustering method that is a variation of the DBSCAN [3] algorithm considering one-dimensional line (trajectories) and speed. In the second step, the algorithm identifies where these potential stops (clusters) are located, considering the candidate stops. In case that a potential stop does not intersect any of the given candidates, it still can be an interesting place. In order to provide this information to the user, the algorithm labels such places as *unknown stops*. Unknown stops are interesting because although they may not intersect any relevant spatial feature type given by the user, a pattern can be generated for unknown stops if several trajectories stay for a minimal amount of time at the same unknown stop. In this case, the user may investigate what this unknown stop is.

Figure 4 illustrates the method CB-SMoT. Considering the trajectory  $T = \langle p_0, p_1, \dots, p_n \rangle$  represented in Figure 4, the first step is to compute the clusters. Suppose that  $T$  has 4 potential stops, the clusters  $G_1, G_2, G_3$  and  $G_4$ , represented by ellipsis. In this example the user has specified 4 candidate stops, identified by the rectangles  $R_{C1}, R_{C2}, R_{C3}$  and  $R_{C4}$ . The cluster  $G_1$  intersects the candidate stop  $R_{C1}$  for a time greater than  $\Delta_{c1}$ , then the first stop of the trajectory is  $R_{C1}$ . The same occurs with the cluster  $G_2$ , considering  $R_{C3}$ , which is the second stop of the trajectory. The clusters,  $G_3$  and  $G_4$  do not intersect any candidate stop. Therefore,  $G_3$  and  $G_4$  are unknown stops.

The two methods cover a relevant set of applications. IB-SMoT is interesting in applications where the speed is not important, like tourism and urban planning. In this kind of application, the presence or the absence of the moving object in relevant places is more important. However, in other



**Figure 4. Example of a trajectory with 2 stops and 2 unknown stops**

applications like traffic management, CB-SMoT, which is based on speed, would be more appropriate.

The output of the *Add Semantics* module are relations of stops and moves in the database. The schema of the stop relation has the following attributes:

```
STOP (Tid integer, Sid integer, SFTname varchar,
      SFid integer, startT timestamp,
      endT timestamp)
```

where:

- *Tid*: is the trajectory identifier.
- *Sid*: is the stop identifier. It is an integer value starting from 1, in the same order as the stops occur in the trajectory. This attribute represents the sequence as stops occur in the trajectory.
- *SFTname*: is the name of the relevant spatial feature type (geographic database relation) where the moving object has stayed for the minimal amount of time.
- *SFid*: is the identification of the instance (e.g. Ibis) of the spatial feature type (e.g. Hotel) in which the moving object has stopped.
- *startT*: is the time in which the stop has started, i.e., the time that the object enters in a stop.
- *endT*: is the time in which the moving object leaves the stop.

In a relational model, the attributes *SFTname* and *SFid* are a foreign key to a geographic relation. Therefore, the stop relation significantly facilitates querying trajectories from a semantic point of view. Queries can be performed considering both spatial and non-spatial attributes of any spatial object that represents a stop. The relation of moves has the following schema, with four attributes more than the stop relation:

```
MOVE (Tid integer, Mid integer, SFT1name varchar,
      SF1id integer, SFT2name varchar,
      SF2id integer, startT timestamp,
      endT timestamp, the_move multiline )
```

where:

- *Mid*: is the identifier of the move in the trajectory. It starts with 1, in the same order as the moves occur in the trajectory.
- *SFT1name* and *SFT2name*: are the names of the spatial feature type in which the move respectively starts and finishes.
- *SF1id* and *SF2id*: are the identifier (feature instance) of the start and end stop of the move.
- *the\_move*: is the set of points that corresponds to the spatial properties of a move.

### 3.3. Transformation

The Transformation module uses as input the tables of stops and moves in the database, generated by the *Add Semantics* module and generates an output file in the format required by a specific mining algorithm or tool. Although each tool can use a specific format, there are two main format types. One, the most used, can be seen as an horizontal type, where each line corresponds to one trajectory and each column corresponds to one stop or move. The other type is a vertical one, where each line corresponds to a stop or move of a trajectory. This second type is mostly used for sequential pattern mining.

Another key issue performed by the *Transformation* module is to generate the output file in the granularity level specified by the user. In fact, the stop and move table is generated in the lowest granularity level (instances of objects for the spatial dimension and timestamp for the time dimension). However, it is almost impossible to find patterns at this granularity level. It is very difficult to some events occur in the same second, for instance several trajectories arriving at home at exactly the same moment. To overcome this problem, in our framework the user can specify different granularity levels, for instance to consider intervals of one hour. This means that one event that occurs at 18:10PM will be considered at the same case as another occurred at 18:20PM. Depending on the application, the time granularity can be year, month, week, day, hour, etc. Analogously, the space granularity can change, including even the semantics of the object. For instance, in the example of Figure 1 the space granularity was the class of the object (Hotel), what will allow that a pattern *from Hotel to Eiffel Tower* could be discovered. In this case, Hotel is at the feature type granularity level and the Eiffel Tower at instance granularity level. If both were at feature type granularity level, the discovered pattern could be *from Hotel to Touristic Place*.

Furthermore, the user can specify what will be considered in the mining step: (i) only the space dimension; (ii) the space and the time of the beginning of the stop or move; (iii) the space and the time of the end of the stop or move; or (iv) the space and the time of beginning and ending of the stop or move.

## 4. Validation and experiments

The proposed framework was implemented in the java programming language in a module called STPM (Semantic Trajectory Preprocessing Module) and tested using Weka as data mining tool and PostGIS as data repository. Weka[4] is a free and open source non-spatial data mining toolkit with several data mining algorithms.

With the information specified by the user, the STPM module connects to the database through JDBC and can execute different tasks. Usually, the first one is to clean the trajectory dataset and put it in the format required by STPM. After that, the user can generate semantic trajectories. To do this, he should supply some information to the program like the method (IB\_SMoT or CB\_SMoT), the spatial feature types of interest (candidate stops) with the respective minimum time duration in order to be considered a stop, etc. Before mining, the last step is to generate an .arff file (the native Weka input format), which can be either in the horizontal or vertical type.

So far, we have tested the prototype with data stored in a Postgresql/Postgis database. We have performed some experiments with real trajectory data collected in the city of Rio de Janeiro. A first experiment was performed considering the districts of Rio de Janeiro and the trajectories. We used the IB-SMoT method and frequent pattern mining to find the districts crossed by a large number of trajectories (minsup=10%) considering time intervals (07:00-09:00, 09:01-12:00, 12:01-17:00, 17:01-20:00, other). Some frequent patterns found are:

```
{Barra[07:00-09:00], Joa[07:00-09:00], SaoConrado[07:00-09:00]}  
      (s= 0.18)  
{Joa[17:01-20:00], SaoConrado[17:01-20:00]} (s= 0.2)
```

The first pattern expresses that 18% of the trajectories cross the districts Barra, Joa and SaoConrado between 7AM and 9AM. The second pattern means that 20% of the trajectories cross Joa and SaoConrado in the period between 5PM and 8PM.

In a second experiment we used the set of streets as background geographic information, and the CB-SMoT method to generate stops. We also used more refined time intervals. The objective was to investigate the streets and periods of slow traffic. An example of an association rule found in this experiment is:

```
ElevadaDasBandeiras[18:01-18:30] →  
AvenidaDasAmericas[18:31-19:00] (s= 0.05) (c=0.58)
```

This rule expresses that 58% of the trajectories with slow traffic at Elevada das Bandeiras between 6PM and 6:30PM, also have slow traffic at Avenida das Americas between 6:31PM and 7PM. This pattern of slow traffic occurs in 8% of the trajectories.

We can observe by the examples above that this framework facilitates the analysis of the obtained results from a user point of view. The output is in a high abstraction level, what we call semantic patterns, in opposition of pure geometric patterns generated by other approaches.

## 5. Conclusions and Future Work

Trajectory data are normally available as sample points, what makes their analysis in different application domains expensive from a computational point of view and quite complex from a user's perspective. A higher abstraction level considering semantics is needed.

This paper presented a framework to preprocess sample point trajectories for semantic trajectory data analysis and mining. By adopting the model of stops and moves we provide to the user aggregated information about space and time. The framework is application and domain independent and it was implemented as an extension of Weka data mining toolkit.

As future ongoing work we are extending the framework in order to consider other methods to add semantics to trajectories.

## Acknowledgements

This work was partially supported by the Brazilian agencies CAPES (Prodoc Program) and CNPq (projects 481055/2007-0 and 573871/2008-6). We would like to thank the Traffic Engineering Company of Rio de Janeiro for the trajectory data.

## References

- [1] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. de Macedo, B. Moelans, and A. Vaisman. A model for enriching trajectories with semantic geographical information. In *ACM-GIS*, pages 162–169, New York, NY, USA, 2007. ACM Press.
- [2] V. Bogorny, P. M. Engel, and L. O. Alvares. Geoarm: an interoperable framework to improve geographic data preprocessing and spatial association rule mining. In K. Zhang, G. Spanoudakis, and G. Visaggio, editors, *SEKE*, pages 79–84, 2006.
- [3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.
- [4] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I. H. Witten, and L. Trigg. Weka - a machine learning workbench for data mining. In O. Maimon and L. Rokach, editors, *The Data Mining and Knowledge Discovery Handbook*, pages 1305–1314. Springer, 2005.
- [5] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In P. Berkhin, R. Caruana, and X. Wu, editors, *KDD*, pages 330–339. ACM Press, 2007.
- [6] OGC. Opendis standards and specifications. Available at: <http://http://www.opengeospatial.org/standards>. Accessed in August 2008, 2008.
- [7] A. T. Palma, V. Bogorny, and L. O. Alvares. A clustering-based approach for discovering interesting places in trajectories. In *ACMSAC*, pages 863–868, New York, NY, USA, 2008. ACM Press.
- [8] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot. A conceptual view on trajectories. *Data and Knowledge Engineering*, 65(1):126–146, 2008.

# A Payload Optimization Technique for Multimedia Visual Cryptography

MOUSSA H ABDALLAH<sup>\*</sup>, ROLA I AL-KHALID<sup>\*\*</sup>, RANDA A AL-DALLAH<sup>\*\*\*</sup>

<sup>\*</sup>Department of Electronics Engineering, Princess Sumaya University for Technology, Amman, Jordan, [moussa@psut.edu.jo](mailto:moussa@psut.edu.jo)

<sup>\*\*</sup>Department of Computer Information System, University of Jordan, Amman, Jordan [r.khalid@ju.edu.jo](mailto:r.khalid@ju.edu.jo)

<sup>\*\*\*</sup>Computer Science Department, Al-Balqa Applied University, Salt, Jordan [randa@bau.edu.jo](mailto:randa@bau.edu.jo)

## Abstract

The technique proposed in this paper is an XOR encryption operation to embed large payload including multimedia information in visual cryptographic color image. The major advantages of the technique are to provide a perfect reconstruction of the image and the increased payload including biometrics, voice, text and images. The algorithm was implemented on large database where it shows its effectiveness when compared to basic watermarking techniques.

*Key-words:* - Visual Cryptography, Multimedia, Information Hiding, Color images

## 1 Introduction

Visual secret sharing (VSS) technique was first proposed by Naor and Shamir in 1994 [1]. The shared secret is an image (such as printed texts, handwritten notes, pictures, etc.), and the VSS scheme provides an unconditionally secure way to encode the shared secret into share images. The decoder is the human visual system. In the VSS technique, several sub pixels in the share are used to represent a pixel in the original secret image, that is, the size of share is expanded. The expansion factor is defined as the size of the share image to the size of the secret image. There have been many published studies [2-6] of visual cryptography. Most of them, however, have concentrated on discussing black-and-white images, and just few of them have proposed methods for processing gray-level [3] and color images.

Rijmen and Preneel [7] have proposed a visual cryptography approach for color images. In their approach, each pixel of the color secret image is expanded into a  $2 \times 2$  block to form two sharing images. Each  $2 \times 2$  block on the sharing image is filled with red, green, blue and white (transparent), respectively, and hence no clue about the secret image can be identified from any one of these two shares alone. Other techniques [8-9] achieved a certain degree of sharing color image information; the drawback is that secret images must be decrypted with heavy computation.

This paper proposed an efficient technique in the multimedia information hiding based on color visual cryptography. This technique will enhance the level of security and the payload by combining cryptography with steganography. Using this approach we embed any multimedia information such as pictures, voices, biometric

fingerprints, personal profiles, medical records and family history in a cover image.

## 2 The Proposed Technique

### 2.1 Hiding Technique

This technique provide three layers of security: First, using a proposed XOR encryption method to encrypt the personal picture and produce three gray shares with gray mask. Then embed the multimedia information in the shares randomly using any steganography technique such as the least-significant-bits (LSBs) method. The resultant shares called multimedia shares. In the LSBs, we use a permutation of bits and pixel locations in which to embed the bits according to private key, so that the hackers may not be able to locate the secret data. Finally we embed the multimedia shares in a cover image using also any steganography technique.

The subtractive CMY model is used in our technique. Each single color based on C, M, and Y can represent 0-255 variations of scale. In the (C, M, Y) representation, (0, 0, 0) represents full white and (255, 255, 255) represents full black.

#### 2.1.1 XOR Method

This paper proposed a new method "XOR" based on visual cryptography to encrypt the secret image. XOR method will produce three gray shares and a gray mask that is used to decrypt these shares. The gray mask generated randomly according to private key, which is used as a signature for encryption and decryption. The gray mask



can be generated in the sending and receiving sides automatically by using the same private key.

In this method, the XOR operation will be used for encryption and decryption. The XOR operation needs computation but the perfect reconstruction of the image is possible.

The color visual cryptography method [10] used digital halftoning technique to convert each color component (C, M, Y) from gray scale to a bi-level, where each pixel has only two possible values: blank or not blank. Halftone technique is used to make the color image suitable for applying original visual cryptography technique. The influence of halftone technique in the reconstructed image (decrypted image) is quality degradation and low contrast. The proposed XOR method is introduced to overcome the effect of the halftone, enhance the quality of the decrypted image and allow visual cryptography to be directly applied on color images. XOR method enhance the quality of the decrypted image by saving  $2^l$  of color levels for each pixel instead of bi-level that is produced by halftoning technique, where  $l$  is the number of bits to represent each pixel.

To encrypt the secret color image using XOR method, transform it into three color components C, M, and Y and reduce the number of the color levels for every pixel ( $S_{ij}$ ) in each color components from  $2^k$  to  $2^l$  according to the following equation:

$$S_{ij} = \text{Round}\left(\frac{S_{ij} \cdot (2^l - 1)}{(2^k - 1)}\right) \dots\dots\dots (1)$$

Where  $1 \leq i \leq N, 1 \leq j \leq M$  and  $k > l$

The quality of the reconstructed image will be enhanced when the number of the bits that represent each pixel increases, but the payload will be decreased. Thus if  $k = l$  then the method can reconstruct exactly the same original image but that will affect the payload drastically. We will reduce the color levels from  $2^8$  to  $2^4$  (where  $k=8$  and  $l=4$ ).

To encrypt every pixel in each obtained color components, a gray mask ( $M_s$ ) is generated that has the same size as the secret image. Each pixel in  $M_s$  contains random number from 0 to  $2^l - 1$  according to a private key ( $KI$ ), as shown in the following equation:

$$M_{s_{ij}} = \text{Round}(\text{Rand}() \cdot 2^l) \dots\dots\dots (2)$$

Where:  $\text{Rand}()$  is a random number generator according to the seed ( $KI$ ),  $0 \leq \text{Rand}() < 1$ .

The  $M_s$  is used to encrypt each color components by XORing the  $M_s$  with each color component to produce three gray shares, a share for every color components. For every encrypted pixel in each share, expand the encrypted pixel into  $n \times m$  block (pixels) by converting the value of the pixel into  $l$  bits string and decomposing each bit in one pixel of the  $n \times m$  block. The block size  $n \times m$  depends on the number of bits  $l$  that represents the pixel value. The block size must satisfy the following equations:

$$n = \text{Round}\left(\frac{l}{2}\right) \dots\dots\dots (3)$$

$$m = 1 - \text{Round}\left(\frac{l}{2}\right) \dots\dots\dots (4)$$

If the original secret image size is  $N \times M$  and the block size is  $n \times m$ , then the size of each gray share will be  $W \times H$ , where:

$$W = n * N \dots\dots\dots (5)$$

$$H = m * M \dots\dots\dots (6)$$

Fig. 1 shows diagrammatically how the XOR encryption method is applied to the colored secret image (Lena).

### 2.1.2 Multimedia Embedding Stage

The multimedia secret information such as personal voice, biometric fingerprint, personal profile and so on is stored in a separate file. For example the personal voice is stored in a wav file, the biometric fingerprint is stored in a bmp file and the personal profile is stored in a text file.

In this stage, the LSBs method is used to embed multimedia secret information in the three gray shares. LSBs is used for two reasons. The first reason is that the LSBs is simple and well known method for embedding information in an image. The second reason is that changing some least significant bits does not result in a noticeable degradation in an image.

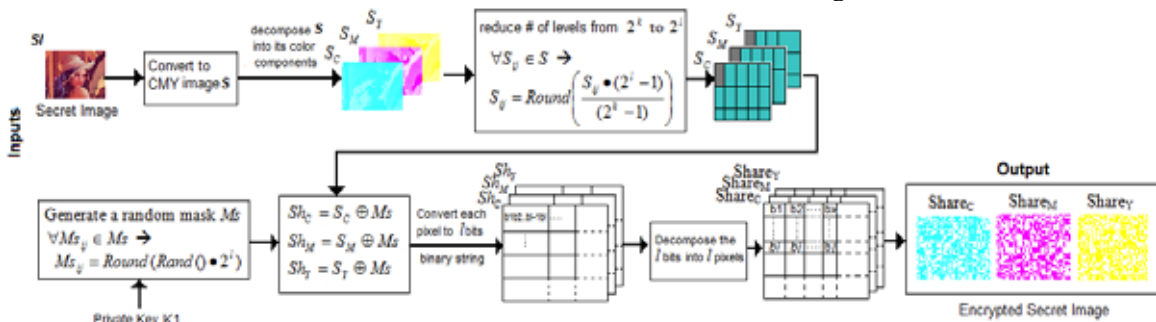


Fig. 1: XOR method

The security of the LSBs method is enhanced through embedding the multimedia information in the shares randomly after permuting all the pixels in the shares using a private key ( $K2$ ). The pixels accuracy for the shares is one bit; we will expand the pixel from 1 bit to 4 bits. The three least significant extra bits are used to embed multimedia information. Before embed the multimedia information, some details about this multimedia information must be embedded such as the number of files to be embedded, the type and size for each file. These details are called header data. The header data is created by converting each byte in the header data into 8 bits and then decomposing the 8 bits into 8 pixels. Select one of the three shares randomly to embed the header data in the first least significant bit, by applying OR operation between the header data bits and the first least significant bits of the selected share. After embedding the header data there are 2 bits free in the share that is selected to embed the header data, and 3 bits free in each of the other two shares. Totally 8 free bits are distributed in the three shares. Use these bits to embed the secret files contents by reading every file byte by byte and convert each byte into a sequence of bits then embed the sequence of bits in the

free bits in the shares. The resultant shares called multimedia shares.

The total size of the embedding data depends on the size of the encrypted secret image. If the secret image size is  $N \times M$  and the block size that is used in the encryption method is  $n \times m$ , then the total number of the bytes that can be sorted in the secret image will be  $n \times N \times m \times M$  bytes.

After embedding the header data and the secret files, restore the original pixel positions of multimedia shares by rearranging the pixels using inverse random permutations according to the private key ( $K2$ ). Fig. 2 shows diagrammatically the embedding stage, where the first least significant bit of  $Share_Y$  is chosen to embed the header data.

### 2.1.3 Final Embedding Stage

In this stage, the multimedia shares obtained from the previous steps is embedded into meaningful color cover image using the LSBs method. This stage embeds the bits of the shares into the four least-significant bit plane of the cover image, which results in a stego-image.

Before embedding the multimedia shares, we must first prepare the shares and the cover as follows:

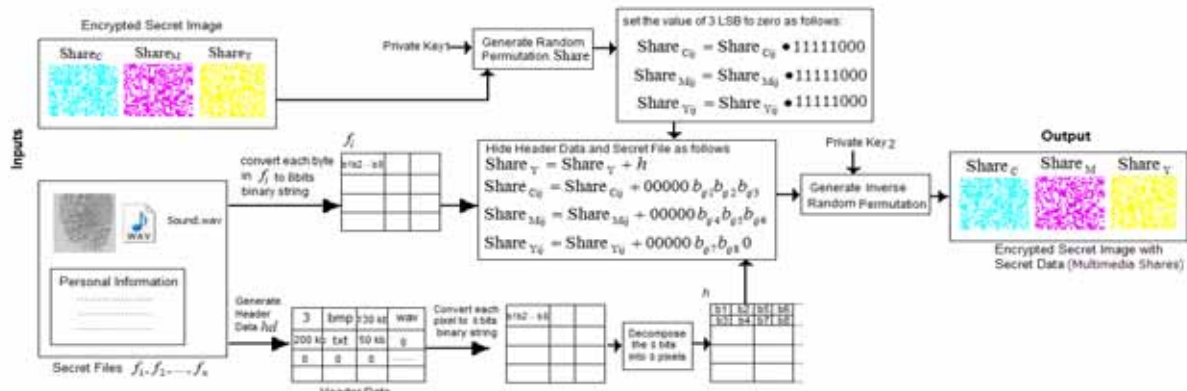


Fig. 2: The Multimedia Embedding Stage.

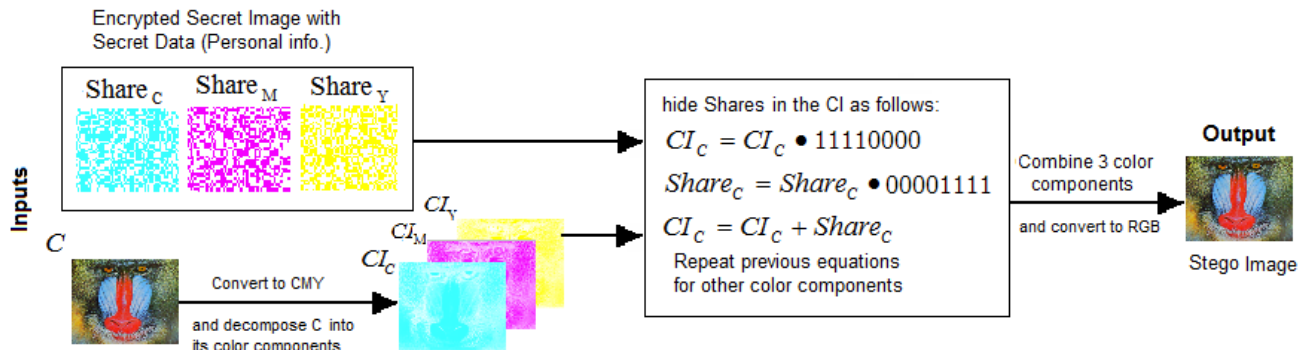


Fig. 3: The Final Embedding Stage

The first step: expand the number of bits for each pixel in the shares from 4 bits to 8 bits and set the value of the 4 most significant bits to zero. The second step: Choose the cover image at least of size  $W \times H$  to be long enough to fit the share (where the size of multimedia shares is  $W \times H$ ).

Once both the multimedia shares and the cover image are prepared, embed each multimedia share into the corresponding color components of the cover image by using OR operation. The resultant image called stego image. Fig. 3 shows the detailed steps of the final embedding stage

Further, to make the LSBs method more robust against casual hackers, use a random combination of bit and pixel permutation techniques according to a private key.

## 2.2 Extraction Technique

In the sending side (hiding technique), the multimedia secret information about the person are embedded in the meaningful cover image by using two keys; ( $K1$ ) to encrypt the secret image and ( $K2$ ) to embed the secret information in the encrypted image. The same two keys must be used in the receiving side to extract the secret image and information from the stego image.

In this technique, the extraction performed by applying the reverse steps of the hiding technique. The multimedia shares extracted first from stego-image, and then the secret information and secret image extracted from multimedia shares. LSB technique is used to extract Multimedia shares from the stego image.

To extract the secret information from the multimedia shares, first a random permutation for every pixel in each multimedia share is applied according to key ( $K2$ ). Second, the header data is retrieved from the share that is selected randomly to embed the header data. Third, the file contents for the  $n$  secret files are extracted by retrieving the bits that contains the secret data from multimedia shares, two bits from the share that embed the header data and three bits from the other two shares. Each eight bits which are retrieved from each three colored pixels (C, M, Y) are combined into one byte. Finally according to the header data, the related bytes are grouped into files.

Since the encrypted secret image was embedded in the 4<sup>th</sup> least significant bit for each pixel in the multimedia shares, the secret image obtained by: First, the 4<sup>th</sup> least significant bits are retrieved and stored in logical shares. Second, the secret image reconstructed by XORing the gray mask  $M_s$  with each logical share. The gray mask  $M_s$  generated randomly according to the private key ( $K1$ ) that was used in the encryption technique as in equation (2). Combine the three new shares together to produce the color secret image. The reconstructed image has  $2^l$  color levels, change the number of levels from  $2^l$  to  $2^k$  by applying the

following equation to each pixel in the image. In our work,  $l=4$  and  $k=8$ .

$$S_{ij} = \frac{S_{ij} \cdot (2^k - 1)}{(2^l - 1)} \dots\dots\dots (7)$$

Where  $l < k$

## 3 Experimental Results

Several experiments have been done, and we would like to describe and discuss the results and its evaluation. Please note that for all the resultant images shown in this paper, they have been scaled down to the same size to fit the page requirements. As a result, there could be a loss in quality.

The experiments are applied on XOR method with  $l = 4$ , where  $l$  is the number of color level. The input of the algorithm is the secret information and the meaningful cover image as shown in Fig. 4.

Note that the size of the cover image satisfies the equations (5) and (6). Fig. 5 shows the results after applying XOR method. Fig. 5(a) is the secret image after applying equation (1) with  $l$  is equal to 4, Fig. 5(b), 5(c) and 5(d) are the generated multimedia shares, Fig. 5(e) is the result of stacking the three shares. Fig. 5(f) is the stego image that hides the stacked image. Fig. 6 shows the final output of the extraction technique. The reconstructed secret image in Fig. 6(a) is the same as the image in Fig. 5(a), and the other secret information in Fig. 6(b), 6(c) and 6(d) are the same as the original secret information.

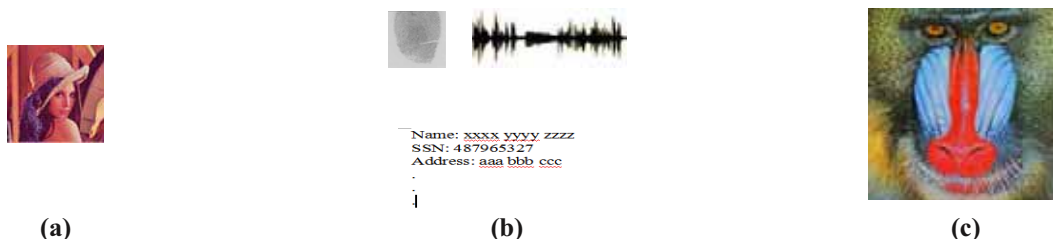
## 4 Payloads and Security Analysis

Data payload is defined as the amount of information it can hide within the cover media. As with any method of storing data, this can be expressed as a number of bits, which indicates the max message size that might be inserted into an image.

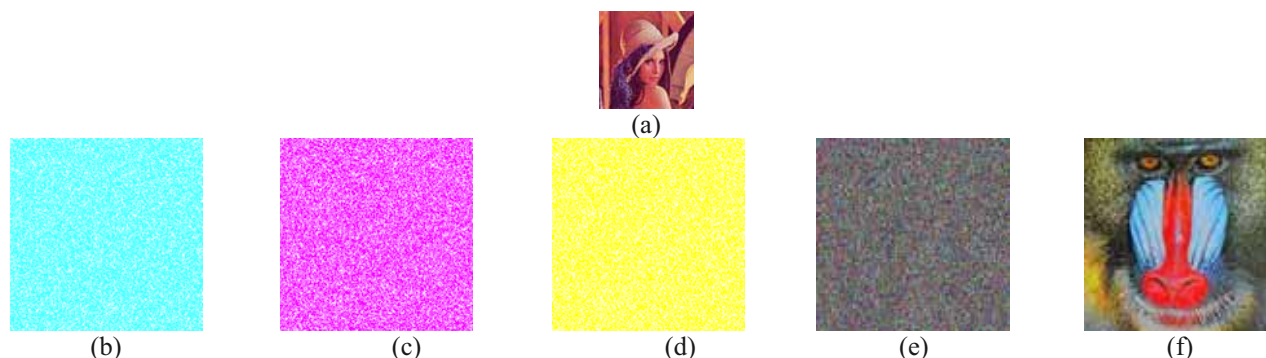
In our technique, we use two cover media to hide secret multimedia personal information, and use the simplest technique LSB method to hide the data in the two cover media. The first cover media is the encrypted secret image that consists of three shares of size  $n \times N \times m \times M$ , where  $N \times M$  is the size of secret image and  $n \times m$  is the size of block that used to encrypt each pixel. In this level of hiding we use three bits for each pixel in each share to hide the data, which leads to data payload  $3 \times 3 \times n \times N \times m \times M$  bits. The second cover media is the meaningful cover image of size  $W \times H$ , where  $W \times H$  satisfy the equations (5) and (6). In this level we use four bits to hide the result of the first level. The data payload in the second cover media is  $3 \times 4 \times W \times H$  bits.

The shares size will depend on the block size that we use in the XOR method. When increase the block size, the shares size will be increased, as consequents the cover

size will be increased. When use the cover media size less than the shares size, the payload will be decreased.



**Fig. 4:** Inputs: personal secret information's and cover image (a) Color secret image of 300 x 300 pixels. (b) Other secret information's such as biometric fingerprint (Bmp file), sample of personal voice (wav file) and personal profile (txt file). (c) Color cover image of 600 x 600 pixels.



**Fig. 5:** Example of XOR method for 16 color levels. (a) Color secret image with 16 color levels. (b) ShareC with secret information. (c) ShareM with secret information. (d) ShareY with secret information. (e) Stacked image (f) Stego image.



**Fig. 6.** The outputs of XOR method after extraction. (a) Secret image. (b) Personal voice. (c) Biometric fingerprint. (d) Personal profile.

Our technique enhanced security, because we use multilevel of security. One of the security levels is encrypt the image randomly to produce three gray shares. The second level is hide secret multimedia data in the random share and in random position. The third level is the secret multimedia data has different type such as wav, bmp and txt. The final level is hiding encrypted image and secret data in meaningful cover image in random position. More levels can be implemented depending on initial specifications and keys.

### 5 Conclusion and future work

The algorithm used in this paper results in perfect reconstruction of the encrypted stego image with larger payload and efficient security layer. As a future work, various attacks including geometric and transformation will be thoroughly studied to see the robustness of the

technique against attacks. The idea of having visual cryptography in watermarking is of interest for data communication and data security.

#### References:

- [1] M. Naor and A. Shamir, Visual Cryptography, in Proceedings of Eurocrypt 1994, lecture notes in Computer Science, vol. 950, 1994, pp. 1–12.
- [2] Ching-Nung Yang, New visual secret sharing schemes using probabilistic method, Vol. 25, No. 4, 2004, pp. 481-494.
- [3] C. Blundo, A.D. Santis, and M. Naor, Visual cryptography for gray-level images, Information Processing Letters, Vol.75, No.6, 2000, pp.255–259.
- [4] T. Hofmeister, M. Krause and H. U. Simon, Contrast-optimal k out of n secret sharing schemes

- in visual cryptography, Theoretical Computer Science, Vol. 240, No. 2, 2000, pp. 471-485.
- [5] G. Ateniese, C. Blundo, A.D. Santis, and D.R. Stinson, Extended capabilities for visual cryptography, Theoretical Computer Science archive, Vol.250, No.1-2, 2001, pp. 143-161.
  - [6] Chang-Chou Lin, Wen-Hsiang Tsai, Visual cryptography for gray-level images by dithering techniques, Vol. 24, No. 1-3, 2003, pp. 349-358.
  - [7] V. Rijmen, B. Preneel, Efficient colour visual encryption for shared colors of Benetton, Eurocrypt'96, Rump Session, Berlin, 1996. Available at: <http://www.iacr.org/conferences/ec96/rump/preneel.ps>
  - [8] R. Youmaran, A. Adler, A. Miri , An Improved Visual Cryptography Scheme for Secret Hiding, in Proceedings of the 23rd Biennial Symposium on Communications (QBSC), Kingston, 2006, pp. 340-343
  - [9] Y.C. Hou, F. Lin, C.Y. Chang, A new approach on 256 color secret image sharing technique, MIS Review, No. 9, December 1999, pp. 89-105.
  - [10] Young-Chang Hou, Visual cryptography for color images, Vol. 36, No. 7, 2003, pp. 1619-1629.

# KNOWLEDGE MANAGEMENT FRAMEWORK FOR CONFERENCE VIDEO-RECORDING RETRIEVAL

Maria Sokhn, Elena Mugellini, Omar Abou Khaled  
University of Applied Sciences of Western Switzerland, Fribourg  
Boulevard de Perolles 80, 1700 Fribourg  
{Maria.Sokhn, Elena.Mugellini, Omar.AbouKhaled}@hefr.ch

## Abstract

*With recording technology becoming easier to use and more affordable due to technological developments, an increasing number of conference talks and academic events are being recorded. However the resulting multimedia data, such as video data, lack rich semantic content annotations. This rich information implicitly conveyed in the large available digital content is today only accessible if considerable effort is made to analyze, extract and create useful semantic annotations. We present in this paper CALIMERA, a knowledge management framework for conference video-recording retrieval that aims to bridge the gap between the implicit knowledge conveyed in the content resources and the explicit representation of knowledge required for efficient multimedia retrieval, access, sharing, and content annotation by communities.*

## 1 Introduction

Recently, an increasing number of conference talks and academic events are being recorded for later access, sharing, use or annotation. However with the exponential growth of digital resources it becomes rather complex to retrieve the appropriate video or video-sequence of a talk due to the lack of rich semantic content annotations associated with those resources. The rich information implicitly conveyed in these resources is only accessible if considerable effort is made to analyze, extract and create useful semantic annotations. Current search engines, for instance, are not able to answer complex a query such as *"Find a sequence of a recorded talk, in 2007, in Italy, where a colleague of professor X, talked about image indexing after the coffee break"* or *"Find a recorded talk of the speaker who wrote the paper Z"* or *"Find a recorded talk of an excellent professor explaining the benefit of ontology use"*. Such queries requires resources to be semantically annotated based on defined concepts re-

lated to the conference domain. For example the concept of *"excellent professor"* in a conference domain may be defined as a person with more than 10 publications, while in an academic or everyday life domain the same concept may be defined differently. Examples cited above illustrate the so-called semantic gap which as defined by [14] (but also other works presented almost similar definitions [33, 19]) *"the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation"*. In order to bridge this semantic gap, information should be indexed according to the users expectations, allowing search engines to find suitable data matching users requirements. Based on this need we designed the framework CALIMERA a a Conference Advanced Level Information Management & Retrieval. CALIMERA framework is targeted for conference video recording. It provides a solution for two main tasks; the knowledge and information management of a conference video recording as well as their retrieval.

Every conference has a life cycle through which rich information is conveyed. We argue that the cOnference High-Level informAtion (OHLA) should be taken into account in video-recordings annotation. OHLA includes the video-recording of the talk with its content extracted information (video segmentation, keywords, topics, etc.), the talk presentation file (ppt, pdf, etc.), the speaker and the audience information (name, organization he/she belongs to it, publications, etc.), the administrative information (conference planning, logistics, etc.), the related demos, the related events, etc.. Such information can be automatically or manually extracted and used to provide rich content based indexing for video-recordings from a user semantic point of view. Manual annotation, is accurate since the description is based on human perception of the semantic content of the video, however it is a labor extensive and time-consuming process especially with the growth of the video collection. Automatic annotation often describes low level content features such as color, shape, etc. Those features lack enough high level semantic information to be useful for users when

searching for data.

In our work we use a mixed approach combining the automatic and the manual approach reducing whenever possible, the burden of a manual annotation. To address the issue of video content description various standards and annotation formats have been developed (MPEG-7, RDF, OWL, etc.). In our work we designed a conference model named HELO (High-level MODEL for cOnference) that models the so-called OHLA. HELO extends and integrates existing conference models.

The paper is structured as follows: in section 2 we introduce the framework CALIMERA, in section 3 we present the conference model HELO, In section 4 we describe the data & metadata management part of the framework. In section 5 we present the related works and finally in section 6 we conclude the paper and present the future work.

## 2 CALIMERA Framework

CALIMERA is an integrated framework aiming to facilitate the retrieval of videos of recorded talks within a conference. CALIMERA framework provides a solution for two main tasks; the knowledge and information management of a conference video recording as well as their retrieval. We argue that the retrieval process is enhanced if the content and the context of a conference during its life cycle are taken into account for describing the resources. Fig. 1 outlines the global view of the framework which is composed of the following modules:

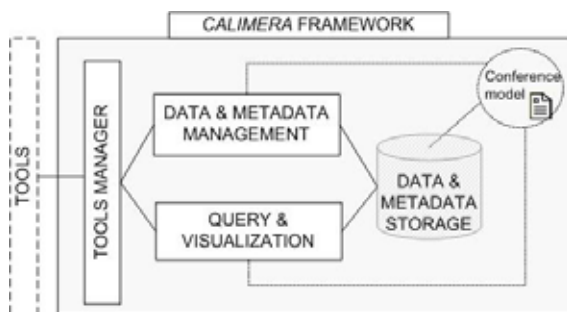


Figure 1. CALIMERA global view

- **Tools manager:** CALIMERA is a tool independent framework. The tool manager allows any user to integrate a tool that may be used for data meta-data management, query & visualization or both (Fig.2). For a proof of concept we integrated four principal tools. SMAC [12]: a tool we developed, to record conference talk and automatically segments the video-recording of this talk based on slide change detections [22, 28]. INDICO [7] is an integrated tool developed by the CERN [3] that manages the administra-

tive part of a conference (such as the conference planning, logistics, etc.). INVENIO [8] and CALISEMA (detailed in section 4.3) are two different tools for video semi-automatic and manual annotation, guided by the conference model, HELO. The two cited tools have been developed in collaboration with the CERN. INVENIO has also a set of modules for automatic features extraction of multimedia data and information indexing.

- **Data & metadata management module** (Fig.2) consists of handling the cOnference High-Level information, such as recording talks, segmenting video recordings, annotating video segments, managing the context information of these talks, etc. A more detailed presentation of this module is given in section 4.
- **Data & Metadata storage** (Fig.2) integrates existing data and metadata formats such as MPEG-7 which is one of the most widely used standard for multimedia description, RDF & OWL, which are a more semantically oriented standards for multimedia description that integrates high level semantic description.
- **Query & visualization module** (Fig.2) queries the data & metadata storage in order to return the video or the set of video sequences of recorded talks the users are seeking for. In order to handle the heterogeneity of annotation standards and formats we designed a format independent interface aiming at querying the heterogeneous data storage [17]. This query engine is based on different concepts such as query analysis, query reformulation, result reasoner, etc..
- **Conference model: HELO** (Fig.1 & 2), is a conference model we designed to model the OHLA. HELO is based on some existing ontologies related to conference domain [5, 24, 21].

## 3 Conference model: HELO

HELO stands for High-level MODEL for cOnference. It models the information and knowledge conveyed during a conference life cycle in order to make best use of the existing information and their eventual relationships. e.g. ‘Professor X’ is not anymore a simple keyword, but an information linked to predefined concepts such as Person, Communities, etc.. We developed the HELO ontology which is a set of different interconnected concepts that have been developed or integrated/extended from existing ones such as the *event* concept [5], the *relationship* [21], the *presentation* [10], the *video-recording structure*, etc. The detailed description of HELO is the key-subject of another paper.

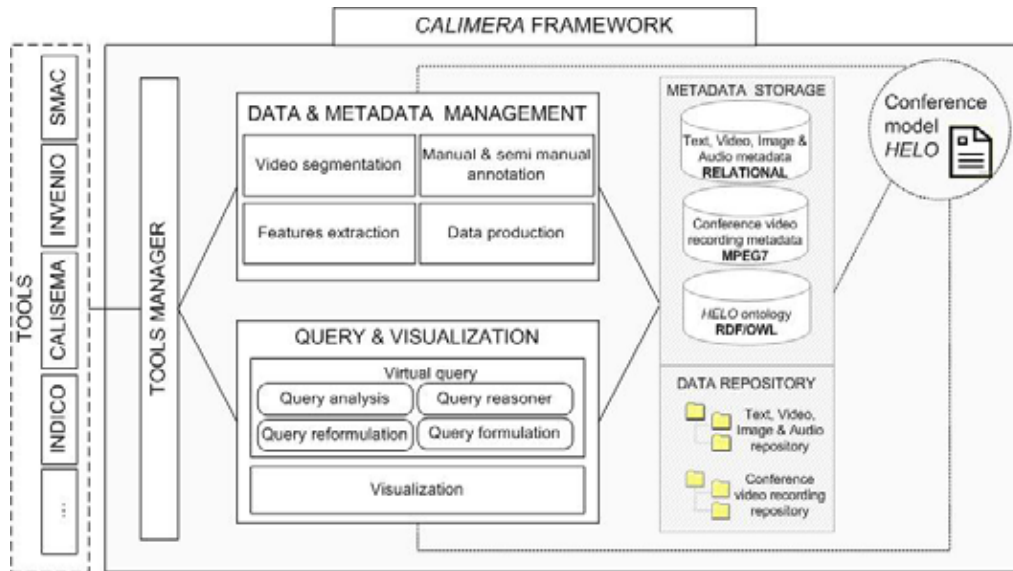


Figure 2. CALIMERA detailed view

We clustered HELO concepts into 7 different views called Scopes

- **Person\_Scope** includes information about people involved in a conference e.g. names, roles, affiliation. It allows users to make queries such as: find the video of the talk where a colleague of the chairman made a presentation.
- **Location\_Scope** contains information about the conference location e.g. continent, city, building, room. It allows users to make queries such as: find the video of the talk that took place in the building A during the conference session Y.
- **Temporal\_Scope** concerns conference planning e.g. starting time, parallel sessions, breaks. It allows users to make queries such as: find the video of the talk that took place in the afternoon in parallel to the talk B.
- **Type\_Scope** lists several categories of conferences e.g. workshop, lecture. It allows users to make queries such as: find a talk given in the academic lecture X.
- **Media\_Scope** gathers all the media information linked to a talk e.g. video recording of the talk, presentation document, papers, books. It allows users to make queries such as: find the video segment of the talk related to this paper.
- **Thematic\_Scope** affiliates a conference to a domain, topic, related events e.g. video indexing, biology. It allows users to make queries such as: find the video part of the talk related to the inauguration of the LHC.

- **Community\_Scope** defines communities such as laboratories, research groups, conferences committee e.g. SEKE program committee, MIT group. It allows users to make queries such as: find the video of the talk where a professor from France university in the SEKE program committee made a presentation.

The detailed description of HELO is the key-subject of another paper. The following section explains the data & metadata management module and describes prototypes for automatic features extraction (SMAC) and semi-automatic video annotation (CALISEMA).

## 4 Data & Metadata management

Different types of information can be associated to a video. Some of them can be automatically extracted, others have to be manually integrated into the video description, or semi-automatically integrated by annotating the video using for example a domain ontology. For instance administrative information of a conference, such as a planning and logistics, usually exists in a digital format (text file, database), thus it can be added to the video description, using an automated extraction process. Conference context description (topic, keywords, communities involved, etc.) and video content description (slide transition, extracted metadata, associated files, etc.) can be extracted using a mixed approach e.g. slide transition can be automatically detected while associating a speaker to a research community requires a manual annotation. CALIMERA combines these approaches. It integrates a set of data management



techniques based on HELO model providing users with efficient and granular search facilities and allowing complex query based on semantic criteria.

#### 4.1 Video segmentation: SMAC

Usually scientific conferences take the form of a series of talks where speakers use slide-based presentations (Microsoft PowerPoint PPT, OpenOffice ODP, Acrobat PDF) displayed as a slide-show during the speech. SMAC capture the speakers slides and records the talks. The synchronization and the linking of those streams are very useful for efficient retrieval and playback. Several prototypes have been proposed to synchronize video recording of a talk with the corresponding images of the slides. Generally, a replay interface allows to playback the talk. Users can select a slide image of the presentation and replay the corresponding video sequence. We developed an algorithm within SMAC project that automatically segments the video recording of the talk and matches each video segment to its corresponding slide. Our approach is guided by metadata information extracted from presentation files and by heuristic hypothesis such as slide show presentations being usually played from the first slide to the last one, or the time passed on each slide is minimum 30 seconds, etc.. As far as we know no prior works such as [31, 26, 15] investigate heuristics hypothesis that exist in the domain of conference presentations. An evaluation of the proposed algorithm has been done using 60 videos of recorded talks (around 50 hours) including animated slides, external demos, non linear slides navigation. Over the 60 videos only two presented an unwanted result. This was due to the speaker slides having a very dark background.

#### 4.2 Video annotation: CALISEMA

Annotation of multimedia information is considered a key issue for efficient information handling. Different multimedia annotation standards have been developed one of the most widely used is the MPEG-7. MPEG-7 supports the description of both low-level and high-level multimedia content features. Low-level features extraction may be automated thanks to several existing algorithms and systems, yet these annotations remain insufficient for a semantic multimedia content description. High level content extraction, which is a richer annotation as stated in the introduction, is a complex and time consuming task requiring manual intervention. To facilitate manual annotation we developed a video annotation tool called CALISEMA. CALISEMA has been developed in collaboration with the university of Athena. It integrates an algorithm manager that allow users to choose the segmentation algorithm they

want to apply on their video such as slide change segmentation for the video recording of talks or shot change detection for video recording of demonstrations, scientific experiments, etc.. CALISEMA has also a segment manager that helps users handling (deleting, adding, or merging) existing segments. Each video segment represented by a keyframe or group of segments can be annotated in CALISEMA using HELO ontology. The description file is exported afterward to either MPEG-7, OWL format or both. Fig.3 shows the CALISEMA interface. The bottom part (1) shows the keyframes of the video recording of the talk. Each keyframes correspond to a slide in the talk presentation. In the left part (2) we view the video sequence corresponding to the chosen slide. Each sequence can be annotated in different ways (top-right part(3)), such as ontology based annotation whose parameters are shown in bottom-right part(4)).

### 5 Related Work

Several research works in the domain of multimedia retrieval have been carried out in the last decade. This section presents the ones which are more relevant to our work.

#### 5.1 Framework

Several frameworks have been designed to handle multimedia information retrieval such as the ones described in [2, 20, 11, 18]. Some of them, such as PHAROS[11], are dedicated for multimedia information retrieval in general. Others, such as COALA [18] for TV news retrieval or ConKMeL [20] for e-learning knowledge management, have been designed based on the domain (news, sport) of the video collection. In fact issues encountered during video processing, such as feature extraction, are addressed differently depending on the video domain. Video analysis and video information extraction techniques can largely differ from a domain to another ( e.g. extracting information from a soccer game would differ largely from extracting information from television news or a conference recording). In our work we decided to develop a framework targeting the scientific conference domain. Our framework extends the VIKEF project approach [2] (a framework for scientific congresses and trade fairs ) by adding several capabilities some of which are the automatic analysis of a conference video recording (such as video segmentation), and heterogeneous data querying.

#### 5.2 Ontology

Several works have been carried out to define an ontological model for conferences and events such as the AKT Reference Ontology [1], the eBiquity Conference Ontology [4]

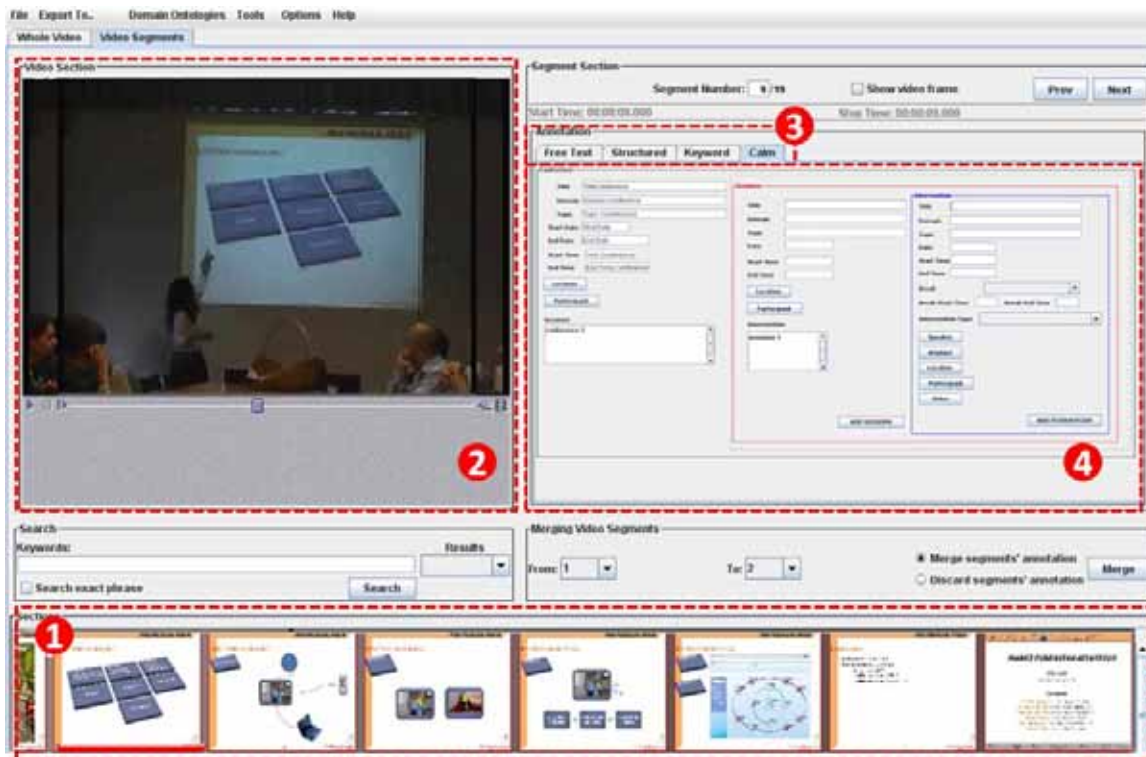


Figure 3. Slide change based video segmentation, HELO annotation

and [21] which describes vocabulary of relations between people. According to a detailed study in [24] these existing solutions lacked the required expressiveness. In ESWC (European Semantic Web Conference) metadata project [6], a more expressive and detailed ontology is proposed, this ontology is called ESWC2006 [5]. It has 6 top-level classes: Artefact, Call, Event, Place, Role, Sponsorship. In contrast to the other ontologies, it models explicitly relationships between people, roles, and events. In our work we integrate the ESWC2006 ontology and extend it (more specifically by adding concepts concerning the video of the conference, such as video conference structure, video sequences, etc.) in order to satisfy users requirements and for efficiently retrieving videos of recorded talks using complex queries.

### 5.3 Tools

#### Video segmentation:

Several industry and research groups work on enhancing the reuse of meetings, conference talks, courses, etc. recording. One of the most important issues to solve is video segmentation. All existing projects are based on segmenting the video according to the slide change of the speaker presentation. Mediasite [13], is a rich media search engine that automatically indexes publicly-available recorded webcasts. Mediasite works on multimodal search

including phonetics, OCR, language models and applying contextual analysis. Jabber [29, 23], is a prototype designed to record, index, and search information generated during multimedia meetings. Marvel (Multimodal Analysis of Recorded Video for E-Learning) [30, 31], is a suite of tools and techniques for the creation of multimedia documents for e-learning taking into account video content indexing. WLAP [26] is a web lecture archive project that aims to implement an electronic archival system for slide-based presentations on the Internet. Mediasite, Jabber, Marvel and WLAP and other projects present interesting results, yet none of them present a completely automated task with a high rate of success in the video segmentation of slide presentation..

#### Video annotation:

Several software programs have been developed to handle multimedia annotation such as Muvino [16] a part of the Vi-TooKi tool kit; Caliph & Emir [25] a project from the Graz University of Technology; M-OntoMat-Annotizer [27] developed using the framework of aceMedia, K-Space Annotation Tool [32] developed at the University of Koblenz-Landau and POLYSEMA MPEG-7 Video Annotator [9] developed at the University of Athens within the Polysema project, etc. Most of these softwares generate MPEG-7 format which is one of the most appropriate multimedia format

while others generate RDF or OWL output.

## 6 Conclusion and future work

This paper presented an end-to-end framework which aims at addressing the existing issues in video indexing and retrieval in the domain of conference. We presented CALIMERA a framework that aims at providing users with efficient and granular search facilities and allows the retrieval of videos of recorded talks of conferences, based on semantic criteria. CALIMERA is based on HELO, a model we conceived to describe the cOnference High-Level information (OHLA) conveyed along a conference life cycle. We presented a video segmentation tool (SMAC) and a video annotation tool (CALISEMA), both based on HELO ontology. As a next step we are currently implementing the designed query engine within query & visualization module.

## References

- [1] Advanced knowledge technologies. <http://www.aktors.org/publications/ontology/>.
- [2] An advanced software framework for enabling the integrated development of semantic-based information, content, and knowledge (ick) management systems. <http://www.vikef.net/>.
- [3] Cern, european organization for nuclear research. <http://www.cern.ch>.
- [4] eBiquity conference ontology. <http://eBiquity.umbc.edu/ontology/>.
- [5] European semantic web conference ontology. <http://www.eswc2006.org/technologies/ontology-content/2006-09-21.html>.
- [6] European semantic web conference project. <http://www.eswc2006.org/>.
- [7] Indico, web application used for scheduling and organizing events. <http://indico.cern.ch/>.
- [8] Invenio, a set of tools for building and managing an autonomous digital library. <http://cdsweb.cern.ch/>.
- [9] Multimedia application supported by semantics. University of Athena. <http://polysema.di.uoa.gr/>.
- [10] Ontology used to manage w3c presentation. <http://www.w3.org/2004/08/Presentations.owl>.
- [11] Pharos: Platform for searching of audiovisual resources across online spaces. <http://www.pharos-audiovisual-search.eu/>.
- [12] SMAC, Smart Multimedia Archiving for Conference. <http://smac.hefr.ch/>.
- [13] Webcasting and knowledge management. <http://www.sonicfoundry.com/mediasite/>.
- [14] A. G. R. J. A.W.M. Smeulders, M. Worring. Content-based image retrieval at the end of the early years, 2000.
- [15] J. J. H. Berna Erol and D.-S. Lee. Linking multimedia presentations with their symbolic source documents: algorithm and applications. pages 498–507. eleventh ACM international conference on Multimedia, 2003.
- [16] K. M. Bszrmenyi L Miller A. Annotation and presentation of content-variations in a webbased search environment for video. pages 67–74. Istvan Simonics (Hrsg.): Technology-enhanced Learning with Ubiquitous Applications of Integrated Web, Digital TV and Mobile Technologie, July 2005.
- [17] E. M. O. A. David G., Maria S. Virtua mea: virtual query architecture for heterogeneous data sources. 2008. Internal publication.
- [18] N. F. PhD thesis, Ecole polytechnique federale de Lausanne, 2003.
- [19] A. P. G. PhD thesis, Queen Mary, University of London, 2006.
- [20] W. H. and A. M. Conkml: A contextual knowledge management framework to support multimedia e-learning, August 2006.
- [21] E. V. Ian D. A vocabulary for describing relationships between people. <http://vocab.org/relationship/>.
- [22] E. M. O. A. Jean Revertera, Maria Sokhn. Video segmentation algorithm for enhanced multimedia synchronization and indexing. 2008. Internal publication.
- [23] R. K. John, K. Accessing multimedia through concept clustering. pages 19–26. ACM SIGCHI, 1997.
- [24] T. H. S. H. J. D. Knud, M. The eswc and iswc metadata projects. pages 802–815. ISWC2007, 2007.
- [25] M. Lux and al. Caliph & emir: Semantic annotation and retrieval. In *Personal Digital Photo Libraries*, pages 85–89. CAiSE '03 Forum at 15th Conference on Advanced Information Systems Engineering, 2003.
- [26] U. of Michigan Media Union. Web lecture archive project. <http://www.wlap.org/>.
- [27] K. Petridis and al. M-ontomat-annotizer, a tool for semantic annotation of images and videos for multimedia analysis and retrieval. 2005.
- [28] J. Revertera. PhD thesis, University of applied sciences of western Switzerland, 2008. Master thesis.
- [29] R. A. W. H. M. M. Rick, K. Four paradigms for indexing video conferences. volume 3, pages 63–73. IEEE MultiMedia, 1996.
- [30] A. B. J.-M. O. Thomas, M. Multimodal analysis of recorded video for e-learning. pages 1043–1044. MM S05, 2005.
- [31] A. B. J.-M. O. Thomas, M. Multimedia scenario extraction and content indexing. pages 204–211. CBMI, 2007.
- [32] P. Wilkins and al. K-space. trecvid, 2007.
- [33] G. L. W.-Y. M. Ying Liu, Dengsheng Zhang. A survey of content-based image retrieval with high-level semantics, January 2007.

# SEPARATING THE SCATTERED CONCERNS: A GRAPH BASED MODEL

Dipankar Majumdar  
B.P. Poddar Institute of Management & Technology  
137, V.I.P. Road, Kolkata – 700052, India  
dipankar.majumdar@gmail.com

Swapan Bhattacharya  
Jadavpur University,  
Kolkata – 700032, India  
bswapan2000@yahoo.co.in

## Abstract

*Separation of Concerns is a well-established concept. SoC is often well-achieved till the Design Phases, but gets difficult in the later phases of the Software Development Life Cycle. This paper makes a graphical approach to address the Code and Maintenance Level SoC. The paper also proposes a formal-model based on relational-algebra to separate out the concerns that remain scattered throughout the source code and make the code complex. This separation will help untangle the tangled code often present in procedural programs.*

## Keywords

Software-Engineering, Separation-of-Concerns, User-defined-Symbols, Relational Algebra.

## 1. Introduction

Addressing complexity is one of the fundamental goals of Software-Engineering. The primary mechanism for addressing complexity has been based on the separation of concerns, allowing distinct concerns to be addressed in relative isolation. Separation of Concerns is quite an established concept. For effective programming and addressing issues like Maintainability, Understandability and Extendibility, SoC is not only desirable but an essential issue. While separation of concerns is generally done in the Design Level, due to limitations of the programming-languages, often the separated concerns tend to intermingle at the implementation level. Some of the concerns get spread throughout the system, scattered through most of the subroutines, while others tend to remain localized to a particular vicinity of the code. The concerns that get scattered in the system tend to pose threat to understandability, maintainability and hence scalability of the system. This paper proposes a graphical representation, namely the ***Symbol Associativity Graph***, for a given procedural source code. Thereafter the paper also proposes a generalized graphical cum mathematical modus-operandi for isolating the scattered concerns. Thereby separating out the un-scattered concerns from the scattered ones, so that they can be placed under separate domains for scalable maintenance.

## 2. Related Work

There has been a wide spectrum of the nature of work that been done in the area of Separation of Concerns. Jonathan Aldrich in his work [7] put forward the challenges that are faced while keeping concerns separated till the implementation level. Adrian Colyer, Awais Rashid and Gordon Blair in their work [8] introduced a set of principles that instruct in the creation of flexible,

configurable, aspect-oriented systems that can help separation of concerns in a software. Y. Smaragdakis and D. Batory in their work [9] proposed how Mixin Layer technique can be used for achieving SoC. N. Bouraqadi in his work [10] proposed the steps behind the usage of the technique: Reflection, for enabling Concern Oriented Programming for achieving SoC while coding. Mark C. Chu-Carroll in his work [11] proposed the technique of file-based program organizations, where the SoC can be addressed and achieved at the implementation level. WalterL.Hursch and C. Videira Lopes in their work [12] identified the major concerns that exist in today's software applications and analyze recent proposals in the literature that address separation of single concerns.

Kim Mens et al in their work [13] proposed the idea of intentional source code views to aid the task software maintenance. They proposed the lightweight abstraction of intentional source-code views as a means of making these conceptual structures more explicit. S. Herrmann and M. Mezini in their work [14] presented their experience with applying multidimensional separation of concerns to a software engineering environment. S. Horowitz et al in their work [16] considered the problem of interprocedural slicing—generating a slice of an entire program, where the slice crosses the boundaries of procedure calls. To solve this problem, they have introduced a new kind of graph to represent programs, called a system dependence graph, which extends previous dependence representations to incorporate collections of procedures (with procedure calls) rather than just monolithic programs. D.L. Parnas in his work [15] presented an analytical survey of the criteria to be used in decomposing a system into modules. Martin. P. Robillard and Gail C Murphy in their work [17] described how they support the evolution of artifacts that refer to the implementation of concerns in a system by combining the ideas of low-level program abstractions, tolerance to inconsistencies, and specialized tool support for inconsistency management. Hafedh Mili et al, in their work [18] proposed a conceptual framework based on a transformational view of software development addressing the issue of Separation of Concerns. Flavio De Paoli in his work [19] presented three examples of separation of concerns and discusses possible solutions. He and Bai [20] propose another aspect mining technique based on cluster analysis. They start from the assumption that if the same methods are called frequently from within different modules, this may be a good indication that a hidden crosscutting concern [6, 23] is present. Marin et al. [21] noticed that many of the well-known scattered and crosscutting concerns [23] exhibit a high fan-in. They propose using a fan-in metric in order to discover these

concerns in source code. Shepherd et al. [24] and implemented as a tool they call *Ophir*, makes use of *program dependence graphs* (PDG) to detect possible scattered and tangled concerns.

Bruntink et al. also make use of clone detection techniques to mine for aspects. In [25, 26], they compare *token-based* [27] clone detection, which is based on a lexical analysis of the source code, with *AST-based* [28] clone detection, which takes the parse tree of the source code into account. Bruntink reports on a refinement of this work [22], in which a number of metrics for the clone classes are described which can be used to filter the results of the clone detection techniques.

Taking the idea from the above work the current paper addresses the problem of identifying and isolating the scattered concerns from a procedural source-code. As put forward by Shepherd et al [24] that scattered and crosscutting concerns [23] may not lie modularized in the code and hence they may be needed to be extracted from the statement level, this paper goes ahead in the way proposing a new graphical model namely the Symbol-Associativity-Graph (SAG) and proposes a formal methodology for identifying and separating out the scattered and tangling concerns, in the form of individual statements, from the other parts of the code.

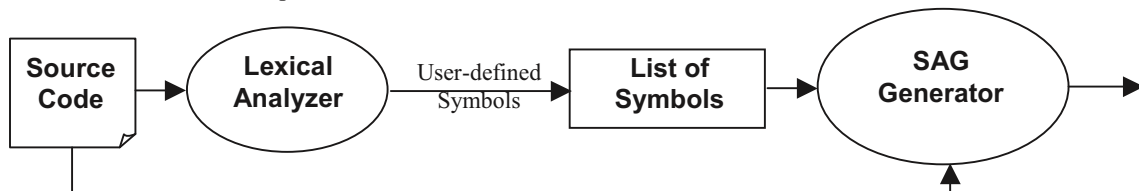


Fig-1: Block Diagram representing SAG Generation Process

### 3. Scope of the Work

Separation of Concerns is a well significant issue to be taken care of in software development. Although it is highly feasible in the earlier stages of SDLC, the issue becomes difficult to follow from the implementation phases onwards. The limitations that exist in the programming languages often pose a threat to the Separation of Concerns. Hence SoC remains un-addressed in some portions of the code. This makes understandability as a result maintenance and scalability difficult. Often concerns that are implemented using scattered variables, scattered codes and functions called

from scattered locations. If these heavily-used variables and frequently-called functions be traced and eliminated, it will be possible to filter out the scattered concerns from the code. This paper proposes a methodology to separate out these tyrannical concerns from the other parts of the program, assuming the fact that any user-defined symbol is declared and defined [13] for and only for a certain purpose or concern. The proposed methodology needs the user-defined symbols along with the source-code as input, which generates the Symbol Associativity Graph (SAG) as output. Fig-1 presents the methodology pictorially.

#### 3.1 Symbol Associativity Graph

Symbol Associativity Graph (SAG) is a graphical model for representing a procedural source code. In this graphical model, stress is laid principally on the associativity of one symbol with the other. The model considers each user-defined symbol as a node/vertex and associativity among them are represented by a directed edge, the direction being from the user to the used. Associativity refers to the 'usage of a variable' or a 'call to a function' from a particular function. Consequently the above implies that the nodes representing variable-names can only act as destination nodes, while function-names can act as both source as well as destination nodes. The edges are not only directed but also carry a string value. The string consists of hyphen-separated numeric values representing the sequence-number of the usage of the destination-node by the source-node in its (source-

node's) domain. Each hyphen indicates the appearance of a compound statement and the following numeric value signifies the sequence number of the usage of the destination-node within that compound statement. The further usage of this sequence-number is beyond the scope of this paper and will be considered for usage in our forthcoming work. After the Graphical Model is built, the next objective is to remove out the frequently used as well as the scattered concerns. Assuming that a particular user-defined symbol is declared and defined for a particular purpose or concern [5], it therefore implies that a widespread usage of a (user-defined) symbol indicates the presence of a scattered-concern. Therefore the nodes of SAG with high 'in-degree' represent its widespread usage. Hence its removal will eventually separate the scattered concerns from the un-scattered ones. The following filtering methodology shows how the objective can be fulfilled.

### 3.2 Filtering Methodology:

Let us assume that a graph  $G(V, E)$ , represents a procedural code, where  $V$ : User-defined Symbols and  $E$ : Inter-Symbol Associativity.

From the above it is clear that a set  $V$  comprises user-defined symbols i.e. data-objects or functional-units and  $E$  represents their edge-label. Representing above graph  $G$  in tabular form, we have a tuple for each link. Our table would have three fields, namely Source-Node, Target-Node and Edge-Label, where for each tuple the Source-Node column would have the entry for the node from which the link originates and the Target-Node column would have the entry for the node on which the link terminates. The third column, namely the Edge-Label would consist of a record of the string that represents the label of the edge under consideration. This way each tuple of our table will represent a single link of the SAG. All the tuples taken together represents the total SAG. We have named our table as Symbol-Associativity-Relationship-Table (SART) shown as Table-1. Since a table logically represents a 'mathematical-relation', we can highly execute relational operations on it. In this paper we have framed a set of ten 'relational-algebra' based queries which when executed sequentially would eliminate the tuples from the SART that are responsible for tangled-code, leaving out the rest. Consequently our left out table, which as previously assumed represents the SAG, and the SAG in turn represents a procedural

program, would consist of source-code devoid of tangled-code and scattered-concerns. Therefore we have converted a code-mining problem to a graphical-problem and the graphical-representation is converted to mathematical-relation such that the 'relational-algebra' can be used to solve the problem thereby obtaining a clean solution. The following list of queries when executed on the Symbol-Associativity-Relationship-Table that represents the SAG in tabular-form, fulfills our objective to separate the scattered concerns from a Graphical-Representation of a Procedural Code. The objectives are to identify the nodes having their in-degree value greater than average in-degree of all nodes. At the same time those nodes are also required to be isolated, which are linked with a large number of *distinct* other nodes. Thereafter these nodes along with their associated edges are to be removed. While removing these nodes along with the associated edges, if their exists any edge/link that represents the condition of a compound-statement, such as a condition or loop whose execution critically depending on the values of any of these nodes (symbol-variables), then the whole block needs to be removed. Our query-based formal-methodology with 'Relational-Algebraic Queries' numbered R9 and R10, caters to that removal activity. In the following query based model, R10 gives the relation representing the graph, which in turn represents the graph devoid of the scattered symbols.

#Relation R1 shows the list of nodes with their usage -frequency.

$$R1 = \rho_{\text{Symbol-Name, Usage-Frequency}} (\text{Target-Node } \mathcal{G}_{\text{Target-Node, count (Source-Node)}} (\text{SART}))$$

#Relation R2 shows the list of nodes with their scattering -values.

$$R2 = \rho_{\text{Symbol-Name, Scattering-Value}} (\text{Target-Node } \mathcal{G}_{\text{Target-Node, count (unique (Source-Node))}} (\text{SART}))$$

# R3 merges the lists represented by R1 and R2 respectively, with their usage-freq. and scattering-value renamed as metric.

$$R3 = \sigma ((\rho_{\text{Symbol-Name, Metric-Value}} (R1)) \cup (\rho_{\text{Symbol-Name, Metric-Value}} (R2)))$$

# Relation R4 computes the average of the usage -frequency and scattering-value for each node.

$$R4 = \rho_{\text{Symbol-Name, Avg-Metric-Val}} (\text{Symbol-Name } \mathcal{G}_{\text{Symbol-Name, avg(Metric-Value)}} (R3))$$

# R5 projects the list of all nodes, each with their usage -frequency, scattering value and their average using Natural Join

$$R5 = \pi_{\text{Symbol-Name, Usage-Frequency, Scattering-Value, Avg-Metric-Val}} (\sigma (R1 \bowtie R2 \bowtie R4))$$

# Relation R6 computes a (self) cartesian-product of R4 with R4, considering only the 'average metric value' and an average of all the 'average metric values'

$$R6 = \rho_{\text{Symbol-Name, Avg-Metric-Val, Avg-Avg-Metric-Val}} (\text{Symbol-Name-X } \mathcal{G}_{\text{Symbol-Name-X, Avg-Metric-Val-X, avg(Avg-Metric-Val-Y)}} (\rho_{\text{Symbol-Name-X, Avg-Metric-Val-X}} (R4)) \times (\rho_{\text{Symbol-Name-Y, Avg-Metric-Val-Y}} (R4)))$$

# R7 shortlists those symbols that have their average metric value higher than the average of all the average metric values.

$$R7 = \pi_{\text{Symbol-Name}} (\sigma_{\text{Avg-Metric-Val} \geq \text{Avg-Avg-Metric-Val}} (R6))$$

# R8 projects the list of nodes with source -node, target-node and edge-label values, where the target-node matches with the list in R7

$$R8 = \pi_{\text{SART.Source-Node, SART.Target-Node, SART.Edge-Label}} (\sigma_{\text{SART.Target-Node} = \text{R7.Symbol-Name}} (\text{SART} \times R7))$$

# Relation R9 projects the list of source -nodes, target-nodes and edge-labels where the edge label starts with any of the edge-labels in R8

$$R9 = \pi_{\text{SART.Source-Node, SART.Target-Node, SART.Edge-Label}} (\sigma_{\text{SART.Source-Node} = \text{R8.Source-Node AND SART.Edge-Label} = \text{R8..Edge-Label}*} (\text{SART} \times R7))$$

\* indicates any group of characters.

# The following relation R10 negates out the list R9 from SART Table.

$$R10 = \text{SART} - R9.$$

The relation R10 represents the resulting SAG representing the same program devoid of the Scattered Concerns

#### 4.0 Case Study

As a Case Study we have taken a C Language source-code of a ‘Two-Player’ ‘Tic-Tac-Toe’ Gaming Software for analysis. The corresponding SAG generated from the source-code in the form of Sub-Graphs is presented in Appendix-I for illustration. The diagrams depict the

viewpoint of each node with its incoming edges only. Since the point of interest of this paper is limited to: the scattering and frequency of usage of any particular symbol, the graph-lets prove to be sufficient enough to provide the said information. The corresponding SART in brief is shown as Table-1.

**Table-1: Symbol-Associativity-Relationship Table**

Source-Node	Target-Node	Edge-Label
CheckVacancy	matrix	1
TestCollinearity	matrix	1
Display	matrix	1-2-1
Set	matrix	1-1
Set	matrix	2
init	matrix	5-1-1-1
...	...	...
...	...	...

According to our model proposed in section 3.2, the relation R5 is shown in tabular form as follows:

**Table-2: Usage-Cum-Scattering Analysis Table**

Serial No	Symbol Name	Usage-Frequency	Scattering-Value	Avg-Metric-Val
1	Matrix	5	5	5
2	fp	9	3	6
3	User1Attempt	6	6	6
4	User2Attempt	6	6	6
5	UserMove	6	6	6
6	DisplayString	7	7	7
7	DisplayMessage	6	4	5
8	WritetoLog	8	5	6.5
9	ClearMessageBuffer	10	5	7.5
10	Init	1	1	1
11	Set	2	1	1.5
12	Display	2	1	1.5
13	TestCollinearity	2	1	1.5
14	CheckVacancy	2	1	1.5
15	CheckValidity	2	1	1.5
16	Terminate	1	1	1

#### 4.1 Calculations & Results:

The Relation R5 presents the Usage Cum Scattering Analysis of all the symbols under consideration. The representation shows clearly that symbols bearing serial nos.: 1, 2, 3, 4, 5, 6, 7, 8 and 9 has the tendency of either getting scattered throughout the source-code or being

frequently used in the source-code. In this paper, the mean ‘Usage-Cum-Scattering Value’ is taken as the Statistical-Central-Tendency for partitioning the symbols on criteria basis.

The Relation R7 according to our proposed model gives the following list of symbols as its result:

- |                 |                  |                       |
|-----------------|------------------|-----------------------|
| 1. matrix       | 4. user2Attempt  | 7. DisplayMessage     |
| 2. fp           | 5. userMove      | 8. WritetoLog         |
| 3. user1Attempt | 6. displayString | 9. ClearMessageBuffer |

According to the symbol listing shown in Table-2, user-defined symbols numbered 1 through 9 are the same as the above list of symbols. While removing the above listed symbols, the care should be taken to remove the compound statements if associated with them. Thus we have relations R8, R9 and R10. Thus, R10 gives the

relation that represents the associativity-relationships of a procedural program devoid of the scattered symbols and the dependencies of any other statements on them. Thus R10 represents a SAG, which again represents a procedural program filtered off from the scattered concerns.

## 5.0 Discussions and Conclusions:

From the above it can be concluded that frequently used and scattered symbols that are principally responsible for scattered problematical concerns can be isolated out from the other concerns that stay localized at particular sections of code. This is to be done principally to handle the two domains: one that of scattered concerns and the other that of un-scattered concerns differently, in different styles of programming paradigm. We advocate the scattered-symbols and their scattered-usage in the code re-coded using the Aspect-Oriented-Programming paradigm, because AOP Paradigm is proved to be very effective [1] in handling scattered-concerns and tangled-code. On the other hand the other localized-concerns may stay back in the original procedural form. Ultimately both can be integrated using the weaver-program [3, 4] used for integrating code-sections in the AOP Paradigm [3, 4]. Our approach is fully practical and scalable for larger-programs as well, so long as procedural-program are concerned. Hence our approach can prove to be a cost-effective for the maintenance phase of SDLC.

## References

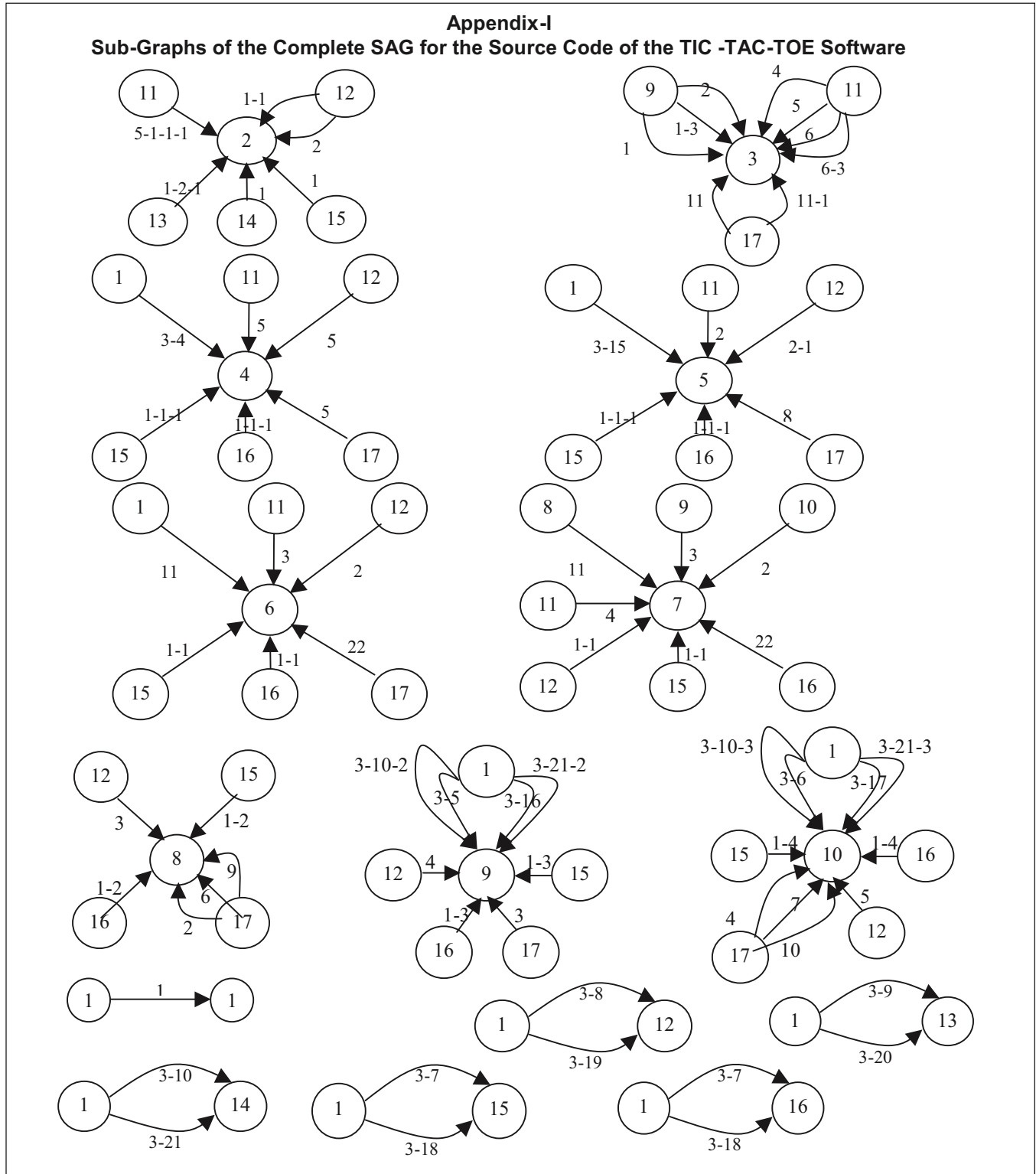
- [1] Jan Hannemann and Gregor Kiczales, Overcoming the Prevalent Decomposition in Legacy Code, *Workshop on Advanced Separation of Concerns (Proceedings)*, International Conference on Software Engineering (May 2001, Toronto, Canada)
- [2] A. Silberschatz, H.F. Korth, S. Sudarshan, "Database System Concepts", 4<sup>th</sup> Edition, Mc Graw Hill, 2002
- [3] Aspect Oriented Programming, <http://en.wikipedia.org>
- [4] <http://www.aspectc.org/>
- [5] Steve McConnell, "Code Complete", Third Indian Reprint: 2006, WP Publishers & Distributors P Limited, pages 255-257
- [6] José María Conejero, Juan Hernández, Elena Jurado, Klaas van den Berg, "Crosscutting, what is and what is not?: A Formal definition based on a Crosscutting Pattern", Technical Report TR28/07. University of Extremadura, 2007
- [7] Jonathan Aldrich, "Challenge Problems for Separation of Concerns", In Proceedings of the OOPSLA 2000 Workshop on Advanced Separation of Concerns
- [8] Adrian Colyer, Awais Rashid, Gordon Blair, "On Separation of Program Families", Families, <http://www.comp.lancs.ac.uk/computing/aop/papers/COMP-001-2004.pdf>
- [9] Yannis Smaragdakis, Don Batory, "Implementing Layered Designs with Mixin Layers", 1998 — In European Conference on Object-Oriented Programming
- [10] Noury Bouraqadi, "Concern Oriented Programming using Reflection", 2000 — In: Workshop on Advanced Separation of Concerns – OOPSLA
- [11] Mark C. Chu-Carroll, "Separation of Concerns: An Organizational Approach", In OOPSLA 2000 Workshop on Advanced Separation of Concerns <http://trese.cs.utwente.nl/Workshops/OOPSLA2000/papers/chucarroll.pdf>
- [12] WalterL.Hursch and Cristina Videira Lopes, "Separation of Concerns", [http://related.reference.kfupm.edu.sa/content/s/e/separation\\_of\\_concerns\\_91681.pdf](http://related.reference.kfupm.edu.sa/content/s/e/separation_of_concerns_91681.pdf)
- [13] Kim Mens, Benard Paul and Sebasti'an Gonz'alez, "Using Intentional Source-Code Views to Aid Software Maintenance", In Proceedings of the International Conference on Software Maintenance, <http://ftp.info.ucl.ac.be/pub/publi/2003/ICSM2003.pdf>
- [14] Stephan Herrmann, Mira Mezini, "PIROL: a case study for multidimensional separation of concerns in software engineering environments", Proceedings of the 15th ACM SIGPLAN conference on Object-oriented.
- [15] D. L. Parnas, "On the criteria to be used in decomposing systems into modules", Communications of the ACM, v.15 n.12, p.1053-1058, Dec. 1972
- [16] Susan Horowitz, Thomas Reps, and David Binkley, "Interprocedural Slicing Using Dependence Graphs", Technical Report SOCS-TR-2005.1 McGill University, Canada, 12 January 2005
- [17] Martin P. Robillard and Gail C. Murphy. "Evolving Descriptions of Scattered Concerns" Technical Report SOCS-TR-2005.1, McGill University, Canada, January 2005
- [18] Hafedh Mili, Houari Sahraoui, Hakim Lounis, Hamid Mcheick and Amel Elkharraz., "Concerned About Separation". In Lecture Notes in Computer Science, Proceedings of the 9th International Conference on Fundamental Approaches to Software Engineering FASE 2006, Vienna, Austria, March 27-28, 2006, pp. 247 - 261.
- [19] F. De Paoli, "Multidimensional Separation of Concerns", In the Proceedings of Workshop on Multidimensional Separation of Concerns in Software Engineering, ICSE-22, IEEE, Limerick (Ireland) June 5-9, 2000
- [20] He, L., Bai, H., Zhang, J., Hu, C.: Amuca algorithm for aspect mining. In: Proceedings of SEKE 2005. (2005)
- [21] Marin, M., van Deursen, A., Moonen, L.: Identifying aspects using fan-in analysis. In: Working Conference on Reverse Engineering (WCRE'04), IEEE Computer Society (2004) 132–141
- [24] Shepherd, D., Gibson, E., Pollock, L.: Design and evaluation of an automated aspect mining tool. In: International Conference on Software Engineering Research and Practice. (2004)
- [25] Bruntink, M., Deursen, A.v., Engelen, R.v., Tourw'e, T.: An evaluation of clone detection techniques for identifying crosscutting concerns. In: International Conference on Software Maintenance (ICSM 2004), IEEE Computer Society Press (2004)
- [26] Bruntink, M., van Deursen, A., van Engelen, R., Tourw'e, T.: On the use of clone detection for identifying crosscutting concern code. IEEE Transactions on Software Engineering 31(10) (2005) 804–818
- [27] Baker, B.: On finding duplication and near-duplication in large software systems. In: Working Conference on Reverse Engineering (WCRE 1995), IEEE Computer Society Press (1995) 86–95



[28] Baxter, I., Yahin, A., Moura, L., Sant' Anna, M., Bier, L.: Clone detection using abstract syntax trees. In: International Conference on Software Maintenance (ICSM 1998), IEEE Computer Society Press (1998).

[22] Bruntink, M.: Aspect mining using clone class metrics. In: 1st Workshop on Aspect Reverse Engineering. (2004)

[23] [http://en.wikipedia.org/wiki/Crosscutting\\_Concerns](http://en.wikipedia.org/wiki/Crosscutting_Concerns)



# Early Analysis of Modularity in Software Product Lines

José M. Conejero, Juan Hernández, Elena Jurado, Pedro J. Clemente and Roberto Rodríguez  
University of Extremadura, Avda. de la Universidad s/n, 10071, Caceres, Spain  
{chemacm,juanher,elenajur,pjclemente,rre}@unex.es

## Abstract

*Software Product Lines has emerged as a new technology to develop software product families based on the combination of a set of common and variable assets. However, in order to combine these assets to build different products, coupling between common and variable parts must be highly reduced. In that sense, crosscutting features make evolution and adaptability of software difficult. In this paper we propose a framework to identify crosscutting features at early stages in order to use aspect-oriented techniques to modularize them and reduce their dependencies. This framework is based on a crosscutting pattern and uses traceability matrices to automatically perform the analysis of crosscutting by means of syntactical and dependencies based analyses. Applicability of the framework is shown by identifying crosscutting features in the MobileMedia product line.*

## 1. Introduction

Software product lines (SPL) have become an emerging trend in software development where products related to a particular domain are created from the combination of a shared set of common and variable software assets [3]. SPL approaches [3] [16] aim at reducing development costs and efforts, while improving the productivity, adaptability and reliability of software systems.

In this setting, feature-oriented modeling techniques analyze commonalities and variabilities among products of a family [6] [10], whereas feature dependency analysis identifies the dependencies among features of a SPL [11]. The effectiveness of a software product line approach highly depends on how well features are managed throughout the development lifecycle [17]: the more independent the assets are, the easier the products may be built [4]. However, features may crosscut each other, making them dependant and reducing thus the flexibility, reusability and adaptability of the SPL assets [17] [4] [9] [12].

Several works have introduced the benefits of using aspect-oriented techniques to deal with crosscutting features, reducing dependencies between them [4] [9] [12]

[13] [15] [17]. However, these proposals only focus on the modeling of variable features in SPL using aspect-orientation. In addition, some of these approaches (e.g. [9] [12] [15]) focus on programming or design stages, relegating the benefits of aspect-orientation to the latest phases of the development.

Besides, as it is stated in [12], common features could be also modeled using aspect-oriented techniques (e.g. aspectual components) if they crosscut to other features. Analogy, variable features need not to be defined always as crosscutting concerns. They may be effectively implemented in modular components if they do not crosscut to other features. Accordingly, although the need of identifying crosscutting features in SPL has been demonstrated in previous works, all the aforementioned approaches just analyze the benefits of incorporating aspect-oriented techniques in SPL and they do not deal with the identification of the crosscutting features (common or variable). Moreover, the incorporation of aspect-oriented techniques at early phases of development improves flexibility and reutilization of the product assets from beginning of the development.

In this context, the major contributions of this paper are twofold. First, it presents a process to automatically identify crosscutting features in SPL requirement artefacts; Second, crosscutting features are then refactored using aspect-oriented techniques, thus complementing other works in the literature such as [1] (Section 3). This is particular useful in the SPL context, where analysis of crosscutting features should be undertaken in early SPL representations. Our identification approach is based on a conceptual framework (Section 2) that is independent of specific requirements and architectural models. Finally, Sections 4 and 5 evaluates the process using a real product line and concludes the paper, respectively.

## 2. A conceptual framework for crosscutting

In [2] we presented a conceptual framework where a formal definition of crosscutting was provided. This framework is based on the study of traceability relationships between two different domains. These domains, generically called Source and Target, could be, for example, features and use cases respectively or, in a

different situation, design modules and programming artefacts. We used the term *Crosscutting Pattern* to denote this situation (see [2]).

The relationship between Source and Target can be formalized by two functions  $f$  and  $g$ , where  $g$  can be considered as a special inverse function of  $f$ .

Let  $f: Source \rightarrow \mathcal{P}(Target)$  and  $g: Target \rightarrow \mathcal{P}(Source)$  be these functions defined by:

$\forall s \in Source, f(s) = \{t \in Target : \text{there exists a trace relation between } s \text{ and } t\}$

$\forall t \in Target, g(t) = \{s \in Source : \text{there exists a trace relation between } s \text{ and } t\}$ .

The concepts of scattering, tangling and crosscutting are defined as specific cases of these functions.

**Definition 1. [Scattering]** We say that an element  $s \in Source$  is scattered if  $card(f(s)) > 1$  (i.e. a source element is related to multiple target elements), where  $card$  refers to cardinality of  $f(s)$ .

**Definition 2. [Tangling]** We say that an element  $t \in Target$  is tangled if  $card(g(t)) > 1$  (i.e. a target element is related to multiple source elements).

**Definition 3. [Crosscutting]** Let  $s1, s2 \in Source, s1 \neq s2$ , we say that  $s1$  crosscuts  $s2$  if  $card(f(s1)) > 1$  and  $\exists t \in f(s1): s2 \in g(t)$  (i.e. a source element is scattered over target elements and where in at least one of these target elements, some other source element is tangled).

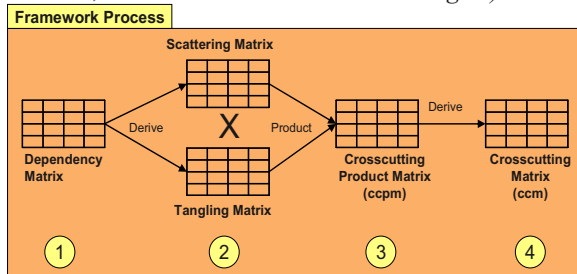


Figure 1. Overview of steps in the framework

In [2] we defined the dependency matrix (a special kind of traceability matrix) to represent function  $f$ . From this matrix, other two matrices (*scattering* and *tangling* matrices) are derived to obtain scattered and tangled concerns of a system. The *crosscutting product matrix* is obtained through the multiplication of scattering and tangling matrices. The *crosscutting product matrix* shows the quantity of crosscutting relations and is used to derive the final *crosscutting matrix* where a cell denotes the occurrence of crosscutting; it abstracts from the quantity of crosscutting.

### 3. Managing early crosscutting features

We have extended our framework defined in [2] with syntactical and dependency based analyses for identifying and managing crosscutting features at requirements level. These analyses allow us to automatically correlate

elements of problem space (source) to elements of the solution space (target). The main steps of our approach are outlined in Figure 2 and summarized as follows:

**(A) Identifying source elements.** We perform a Feature-Oriented Analysis to obtain the main features of the product family including both common and variable assets (Figure 2-(1)). We also search for evidences of non-functional concerns (NFC) that appear in the requirement documents (Figure 2-(2)).

**(B) Identifying target elements.** In this phase we model the requirements (Figure 2-(3)) using use cases.

**(C) Build the dependency matrix.** Taking features (also including the NFC) and use case artefacts as source and target respectively, we establish the trace relations between them (function  $f$  defined in Section 2). These trace relations are automatically established by means of syntactical (Figure 2-(4)) and dependencies based (Figure 2-(5)) analyses.

**(D) Identification of crosscutting by matrix operations.** Applying several matrix operations (Figure 2-(6) and Figure 2-(7)), shown in Section 2, the crosscutting features (Figure 2-(9)) at requirements level are obtained.

**(E) Aspect-oriented refactorization.** Finally, the crosscutting features identified are modeled using aspect-oriented techniques (Figure 2-(8 and 9)).

#### The example: the MobileMedia product line

In order to illustrate the process, we apply it to a well-know case study, the MobileMedia product line [7]. The MobileMedia is a product line system built to allow the user of a mobile device to perform different operations such as visualizing photos, playing music or videos and sending them by SMS. The system has been built as a product line in 8 different releases (from 0 to 7). In this section we use a particular release (release 3) to illustrate the process explained in this paper. This release includes the functionality to manage albums and photos and some other optional features like sort photos by frequency, edit labels and set favourite photos. This release was selected because it presents some variable features and includes the presence of non-functional concerns. Nevertheless, the release is simple enough to not complicate the explanation of the process (in Section 4 we show the application of the process to all the releases). In next subsections, we explain each activity of the process represented in Figure 2

### 3.1. Feature-Oriented Analysis

Feature-Oriented Domain Analysis (FODA) [10] is a domain analysis technique which allows the developer to improve the understanding of software requirements. In this section, we focus on the feature model for the MobileMedia (Figure 2-(1)). Since the system has been used in previous analyses, we utilize the same feature model used by the original authors [7] (shown in Figure

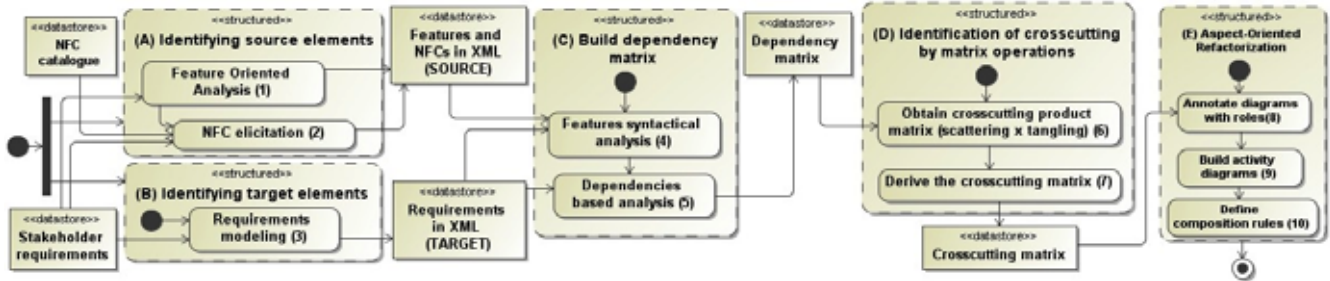


Figure 2. Main phases of the identification of crosscutting features

3). Note that variability between products is mainly concentrated in the possibility of sorting photos, setting the favourites and editing labels.

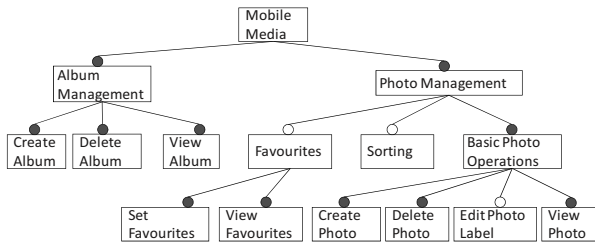


Figure 3. Features model for MobileMedia

We represent the features in a XML file (Figure 4) with `<feature>` tags. The subelement `<keyword>` represents the words that we use to relate the source element with elements of the target domain (explained later on).

```
<?xml version="1.0" encoding="UTF-8"?>
<FeaturesFile>
  <feature id="f4" name="Sorting">
    <description>Feature related to ... </description>
    <stakeholder>
      <user>user</user>
    </stakeholder>
    <keyword>Sort</keyword>
    <keyword>Count</keyword>
    <keyword>Frequency</keyword>
  </feature>
  <nfc id="c1" name="Persistence">
    <description>Way of storing ... </description>
    <stakeholder>
      <user>developer</user>
    </stakeholder>
    <keyword>store</keyword>
    <keyword>retrieve</keyword>
  </nfc>
</FeaturesFile>
```

Figure 4. Feature and NFC in XML format

### 3.2. Non-functional concerns elicitation

Secondly, we identify the non-functional concerns involved in the product line (Figure 2-(2)). This analysis is performed using a catalogue (in XML) where common non-functional concerns are presented and related to different words that usually describe them in requirements documents. We use these words to analyze the stakeholder requirements so that non-functional concerns are identified when one of these words appears in the requirements documents.

Using the requirements of the system presented in [7], we have identified some non-functional concerns: *Persistence* and *Error Handling*. *Persistence* is present in

the system since the photos or any media file must be stored in the mobile memory. *Error Handling* is added in release 1 and it is included in the rest of releases (from 2 to 7). Then, the elements of the source domain in the MobileMedia are the features *Album*, *Photo*, *Label*, *Sorting* and *Favourites* and the non-functional concerns *Persistence* and *Error Handling*. The non-functional concerns are also represented in the XML features file (`<nfc>` tags in Figure 4).

### 3.3. Requirements modeling

In this activity we build the first representation of the system using UML use case diagrams (Figure 2-(3)).



Figure 5. Use case diagram for MobileMedia system

The use case diagram (Figure 5) is stored in XML format (see Figure 6).

```
<packagedElement xmi:type="uml:UseCase"
xmi:id="1221839017656_481435_864" name="View Photo">
  <include xmi:id="1222084791125_970342_2817"
  addition="1222084764703_704034_2805"/>
</packagedElement>
...
<packagedElement xmi:type="uml:UseCase"
xmi:id="1222084764703_704034_2805" name="Count Photo"/>
```

Figure 6. XML generated from diagram of Figure 5

### 3.4. Build the dependency matrix

The trace relations between elements of source (features and NFCs) and target (use cases artefacts) domains are represented by means of the dependency matrix. A cell with one denotes that the target element of this column contributes to the source element of the corresponding row. The dependency matrix is automatically built by means of two analyses: Syntactical

(Figure 2-(6)) and Dependencies based (Figure 2-(8)). The inputs of the phase are the XML files generated in previous phases whilst the output is the dependency matrix built.

### Syntactical analysis

In this activity we discover trace relations between features and non-functional concerns (source elements) and artifacts of the use case diagram (target elements). We relate these two set of elements through a syntactical analysis based on the similarities among the identifiers of these elements. The values of the different <keyword> tags of the features file are totally or partially compared (using the whole word or just the morpheme) with the attribute name of the <packagedElement> tags (in the use cases XMI file). For instance, we can relate the feature *Insert* to a use case with the name *Insert file* but also *Insertion of files*.

In order to compare the identifiers defined in the XML files, we use the XQuery language [18]. Using different queries, we obtain the relations shown in higher part of Table 1 (cells with 1 and in light grey background).

### Dependencies based analysis

Next we search for indirect dependency relations between source and target elements by analyzing the existing relations between the elements of the target domain as follows: *if s1 is related to t1, and t2 depends on t1 then s1 is related to t2*, being  $s1 \in \text{Source}$ , and  $t1, t2 \in \text{Target}$  (a special kind of transitivity relation). Clearly, the <<include>> relations in the use case diagram represent such dependencies between target elements. We do not use <<extend>> relations because they represent a specialization and not a dependency (the extended use case does not really depends on the use case which extends it).

Table 1. Dependency matrix after the analyses

		Use cases														
		Add Album	Delete Album	Add Photo	Delete Photo	View Photo	View Album	Provide Label	Store Data	Remove Data	Retrieve Data	Edit Label	Count Photo	View Sorted Photos	Set Favourite	View Favourites
Features	Album	1	1				1									
	Photo			1	1	1										
	Label	1		1				1				1				
	Sorting					1							1	1		
	Favourites															1
NFC	Persistence	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Error Handling	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

We can see in Figure 5 that there are different <<include>> relations. For instance, there are several use cases that include the functionality of the *Store Data* use case. Since the *Store Data* use case is contributing to *Persistence* (see Table 1), we relate all the use cases which include the *Store Data* use case to the *Persistence*

NFC. Indirect relations are shown in dark grey background in Table 1.

The application of the dependencies based analysis is also automatically done by means of analyzing the XMI file (see Figure 6). The <include> tag has an attribute called *addition* (pointing out to the included use case). We just need to search the identifier of the included use case in the rest of file (e.g. *View Photo* use case has an <include> tag with the *addition* attribute pointing out to *Count Photo*). A simple Java tool allows us to process the XMI file.

In some cases, there are some NFC that do not explicitly appear in the use case diagram. The process helps to identify these situations since the dependency matrix would have a null row (without any mapping). For instance, in *MobileMedia*, *Error Handling* is not explicitly present in the use cases so that it would not be related to any use case. These situations may be solved by manually reviewing the use case descriptions. We identified that all the use cases were constrained by the *Error Handling* NFC (then we added the mappings shown in black cells in last row of Table 1). In other cases, the NFC could be related to an architectural or hardware decision (e.g. using a particular hardware platform to deal with performance or an architectural pattern to improve reusability) and thus it would not constrain the software modularity.

### 3.5. Identification of crosscutting by matrix operations

Using the dependency matrix and the conceptual framework introduced in Section 2, we derive the final *crosscutting matrix* (Figure 2-(7)). A cell with 1 indicates that the element of this row is crosscutting to the element of the corresponding column (see Table 2).

Table 2. Crosscutting matrix for the MobileMedia

		Features					NFC		
		Album	Photo	Label	Sorting	Favourites	Persistence	Error Handling	
Features	Album				1		1	1	M
	Photo			1	1		1	1	M
	Label	1	1				1	1	V
	Sorting		1				1	1	V
	Favourites						1	1	V
NFC	Persistence	1	1	1	1	1		1	
	Error Handling	1	1	1	1	1	1	1	

Table 2 firstly confirms what intuition perceives: NFC *Persistence* and *Error Handling* are the elements which crosscut to more features. This table also shows how Mandatory features (Album and Photo) may crosscut Variable features (e.g. Label or Sorting) and vice versa. This situation suggests the use of aspect-oriented techniques to isolate and refactor NFCs and crosscutting

features. Isolating a certain crosscutting feature removes the crosscutting dependencies between features. However, if two given features A and B are crosscutting each other, what feature should be refactored, A or B? In Section 4, we show how to take such decisions by an empirical analysis driven by a set of concern metrics [5].

### 3.6. Aspect-oriented refactorization of crosscutting features

In [14], the authors present a method to modularize volatile concerns at requirements level by aspect-oriented techniques using Pattern Specification [8]. We have adapted this technique to refactor the crosscutting features in the MobileMedia product line. The use cases implementing the crosscutting features are marked using the special symbol “|” (Figure 2-(8)). A new relation <<crosscut>> is added to the use case diagram which relates use cases implementing crosscutting features to those which are considered as the base functionality (see Figure 7).



Figure 7. Use case diagram marked

Once crosscutting features have been isolated in the marked use case diagram, different products may be built simply changing features using composition rules. By these composition rules we may compose different activity diagrams. As an example, in Figure 8 and Figure 9 we show two activity diagrams which represent the main flows of the *View Photo* and *Count Photo* use cases (Figure 2-(9)). These activity diagrams are composed using the composition rule shown in Figure 10 (Figure 2-(10)). The addition or removal of the *Sorting* feature is as easy as applying or not the composition rule in the system, respectively. We could also use a different way of sorting photos just using a different composition rule and thus composing different activity diagrams.

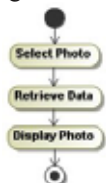


Figure 8. View Photo activity diagram



Figure 9. Count Photo activity diagram

Compose ViewPhoto with CountPhoto  
 Insert Retrieve observed times after Display Photo  
 Figure 10. Composition rule for the diagrams

## 4. Evaluation and discussion

In this section we have analyzed the crosscutting relations in the MobileMedia product line (including the 8 different releases). We have used a set of concern-oriented metrics [5]. These metrics are automatically calculated using our dependency matrix (Section 3.4). Since the metrics are generic (not tied to any specific development domain or level), we have used them in the SPL context relating features and use cases as source and target domains respectively. In Figure 11 we show a graphic showing the Degree of Crosscutting metric of the different features throughout all the releases (see the whole concern-oriented metrics in [5]). The closer to 1 the values obtained for this metric are, the more crosscutting a feature has. We use the metric to decide the features or concerns that should be refactored. Then it may be used as an *oracle* to support developer’s decisions.

The analysis of the results shows the following evidences: i) *Persistence* and *Error Handling* concerns present the higher Degree of Crosscutting in all the releases. ii) Some variable features (e.g. *Label*) also present a Degree of Crosscutting higher than the rest of features in all the releases. Note that any change in *Label* feature (including the removal) implies the modification of all the features crosscut by it (actually the artefacts implementing these features). iii) Some mandatory features, such as *Photo* also crosscut to other features. These results show the evidence about the need of using aspect-orientation to modularize these features, reducing dependencies between them and improving, thus, the reutilization of the software assets.

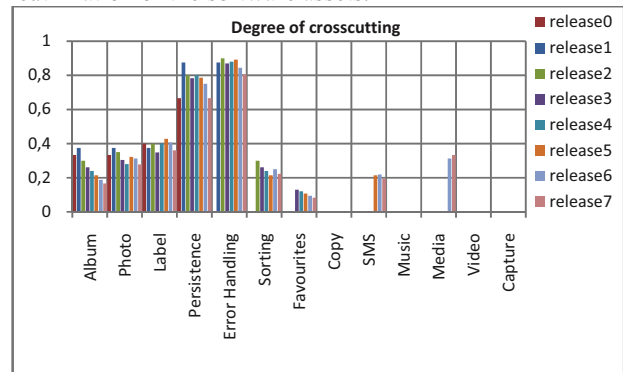


Figure 11. Degree of crosscutting for MobileMedia

The metrics allow us to focus on the features with a higher Degree of Crosscutting. Accordingly, and as we mentioned before, the values obtained for the metric suggest the following: i) NFCs *Persistence* and *Error Handling* should be refactored to avoid crosscutting dependencies with mandatory and variable features. ii) As *Label* and *Photo* are crosscutting each other but *Label* presents a higher Degree of Crosscutting, the *Label*

feature should be refactored, as it was shown in Section 3.6.

In general, we could say that the refactorization of variable features allows a better reutilization of the core assets in order to build other products (just adding different aspects to the systems). However, in cases where a mandatory feature crosscut other mandatory features, a change in it implies the modification of the features crosscut, thus they should be also modeled using aspects.

## 5. Conclusions

It has been demonstrated in the literature that AOSD helps to reduce crosscutting dependencies between assets, improving flexibility, configurability and reutilization of SPL assets. In order to introduce the benefits of AOSD in SPL, we have presented a process to identify the crosscutting features at early stages so that they may be isolated at the very beginning of the software development process. Then, the process presented allows us to improve the product line modularity getting as result an important improvement in configurability and reutilization of the family products.

In addition, the process presented is semi-automatically applied using features and requirements models. Then, the results obtained may be linked to other aspect-oriented design approaches (e.g. [1]) which define the system UML models. These models can be used to generate the product family.

## Acknowledgements

This work has been supported by MEC under contract: TIN2008-02985.

## References

- [1] Alférez, M., Kulesza, U., Sousa, A., Santos, J., Moreira, A., Araújo, J. and Amaral, V. (2006). *A Model-Driven Approach for Software Product Lines Requirements Engineering*. In the 20th International Conference on Software Engineering and Knowledge Engineering, San Francisco Bay, USA
- [2] Berg, K. van den, Conejero, J. and Hernández, J. (2007) Analysis of Crosscutting in Early Software Development Phases based on Traceability. *Transactions on Aspect-Oriented Software Development III*, LNCS 4620, pp. 73–104, Springer-Verlag. Special issue on Early-Aspects.
- [3] Clements, Paul & Northrop, Linda (2002). *Software Product Lines: Practices and Patterns*. Boston, MA: Addison-Wesley
- [4] Colyer, A., Rashid, A. and Blair, G. (2004). On the Separation of Concerns in Program Families. *Lancaster University Technical Report Number: COMP-001-2004*
- [5] Conejero, J., Figueiredo, E., Garcia, A., Hernández, J. and Jurado, E. (2009) *Early Crosscutting Metrics as Predictors of Software Instability*. To appear in 47th International Conference Objects, Models, Components, Patterns (TOOLS Europe). Zurich, Switzerland
- [6] Czarnecki, K., Eisenecker, U. (2000). *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Reading, MA
- [7] Figueiredo, E., Cacho, N., Sant’Anna, C., Monteiro, M., Kulesza, U., Garcia, A., Soares, S., Ferrari, F. Khan, S., Filho, F., Dantas, F. (2008). *Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability*. In proceedings of the 30th International Conference on Software Engineering (ICSE), Leipzig, Germany
- [8] France, R., Kim, D., Ghosh, S. and Song, E. (2004). A UML-Based Pattern Specification Technique. *IEEE Transactions on Software Engineering*, Volume 30(3)
- [9] Griss, M. (2000) Implementing product-line features by composing aspects. In proceedings of *First International SPL Conference*, pp. 271–288, Denver, USA
- [10] Kang, K., Cohen, S., Hess, J., Novak, W. and Spencer A. (1990). Feature Oriented Domain Analysis (FODA). Feasibility Study. *Carnegie Mellon University Technical Report CMU/SEI-90-TR-21*
- [11] Lee, K., Kang, K., Kim, M. (2004). Feature Dependency Analysis for Product Line Component Design. In the 8th International Conference (ICSR), Madrid, Spain. LNCS 3107, Springer, pp. 69-85
- [12] Lee, K., Kang, K., Kim, M. and Park, S. (2006). Combining Feature-Oriented Analysis and Aspect-Oriented Programming for Product Line Asset Development. In proceedings of the *10<sup>th</sup> International SPL Conference*, Baltimore, USA
- [13] Loughran N., Sampaio, A. and Rashid, A. (2005). From Requirements Documents to Feature Models for Aspect Oriented Product Line Implementation. In proceedings of *workshop on MDD for Product Lines at MODELS 2005*. Montego Bay, Jamaica
- [14] Moreira, A., Araujo, J. & Whittle, J. (2006). Modeling Volatile Concerns as Aspects. In *18th Conference on Advanced Information Systems Engineering*. LNCS 4001/2006: 544-558. ISBN: 978-3-540-34652-4, Luxembourg.
- [15] Morin, B., Barais, O. and Jézéquel, J.M., (2008). Weaving Aspect Configurations for Managing System Variability. In Proceedings of *Second International Workshop on Variability Modelling of Software-intensive Systems (VaMoS’08)*, Essen, Germany
- [16] Pohl, K., Böckle, G. and van der Linden, F. (2005) *Software Product Line Engineering: Foundations, Principles and Techniques*. Berlin, Germany. Springer.
- [17] Voelter, M. & Groher, I. Product Line Implementation using Aspect-Oriented and Model-Driven Software Development. In proceedings of the 11th International SPL Conference, 2007
- [18] XQuery 1.0 (2007): *An XML Query Language*. W3C Recommendation, 23 January 2007. <http://www.w3.org/TR/xquery/>

# MD-JPA profile: A model driven language for Java persistence<sup>1</sup>

**Alexandre Torres**

Instituto de Informática, UFRGS,  
Porto Alegre, RS, Brazil  
email atorres@inf.ufrgs.br

**Renata Galante**

Instituto de Informática, UFRGS,  
Porto Alegre, RS, Brazil  
email galante@inf.ufrgs.br

**Marcelo S. Pimenta**

Instituto de Informática, UFRGS,  
Porto Alegre, RS, Brazil  
email mpimenta@inf.ufrgs.br

## Abstract

*The model driven development (MDD) approach proposes models playing the main role on system development. However, there is not a consensual notation to model persistence based upon object relational mapping frameworks: while UML lacks specific resources for persistence modeling, the entity-relationship model does not make reference to the dynamic concepts existing in UML. This paper proposes MD-JPA, a UML profile for persistence modeling based on the well-known Java Persistence API (JPA), pursuing the modeling of transient and persistent elements in a more coherent and integrated way. This paper specifies the main characteristics of MD-JPA as well as the way that models that adopt such profile can be then turned into Java implementation by the use of model transformations proposed on a MDD approach.*

## Keywords

Persistence, UML, MDD, JPA, Database.

## 1. INTRODUCTION

The Unified Modeling Language (UML) was born from the object oriented paradigm [13] and has rapidly become a standard notation for software modeling, despite its lack of resources for data modeling (persistence) [1]. The entity relationship (ER) model [3] and derivatives are still employed as the main artifacts for conceptual database modeling, and both UML and ER models have still coexisted well in the software engineering processes.

The Model Driven Development (MDD) proposes that models take on the main role on the system development process, replacing parts or the whole process of software coding [2]. By using MDD approaches the information represented in a set of models should be coherent, integrated, and computable so that automatic transformations could turn models into executable system [9]. Using separated UML and ER models turned out to be a problem for transformation construction: there is neither integration between elements nor a common meta-model. Besides, there is the object-relational impedance mismatch problem [1].

Object-Relational Mapping (ORM) frameworks address the impedance problem on software implementation level [1]. The Java Persistence API (JPA) [4] is a widely adopted specification of ORM for the Java platform, enabling the representation of both object oriented concepts and relational database specification with annotated Java programs.

However, the ORM modeling lacks a modeling language that can express its concepts.

This paper proposes MD-JPA, a UML profile that enables the representation of database and software structures using the JPA standard, within the MDD approach. With such extension, models may be used as artifacts in the software development cycle and as a source for transformations that aim at the generation of pieces of the system.

The MD-JPA profile is validated by a set of transformations that can turn well formed models into JPA compatible implementation. The main contribution of this paper is a UML profile that implements aforementioned extensions and can be applied to models created by standard UML design tools. Moreover, a set of constraints checks models detecting incompatible designs with ORM, reducing the round-trip effect of adapting models to development needs.

The remaining sections of the paper are organized as follows. Section 2 presents related work on persistence modeling, ORM, and transformations. Section 3 specifies the MD-JPA modeling elements and section 4 discuss the constraints used to validate models. Section 5 validates the profile by presenting transformations that take models producing JPA implementation. The last section brings the final remarks and future work.

## 2. Related Work

The agile database modeling [1] is a well known proposal for database modeling using UML extensions. It is mainly based upon the class diagrams for representing data models with a set of model, class, and property stereotypes, allowing the creation of models in 3 abstraction levels: conceptual, logical, and physical.

The conceptual and logical models focus on the representation of the ER concepts such as entities, relationships, and attributes, by annotation of classes, properties, and associations. The physical model represents database concepts such as tables, columns, views, and foreign keys. The result is an extension that successfully allows the data modeling with UML but does not tackle the integration of object oriented and relational concepts.

The Object Management Group (OMG) has an underway proposal for data modeling representation. Again the focus seems to be the representation of ER concepts with UML, aided by database reverse engineering, providing a wider range of concepts to allow XML representation [10].

<sup>1</sup> Partially financed by project CNPq/CT-INFO 550891/2007-2 and CNPq/TISHE - Edital Universal 02/2006



Both proposals enable the use of UML for data modeling, therefore allowing the construction of transformations over a common meta model. However, to successfully tackle the object-relational impedance mismatch problem, the data and object modeling should be more integrated and coherent [1], hence facilitating the visualization of the relations between tables and classes that access those tables.

UML extensions related to persistence, approaching relational database operations [18] and multidimensional modeling in data warehouses [19], are focused in the modeling details of data representation. The object oriented implementation of such data representation with ORM is not part of those works.

Similar works were proposed for modeling object oriented databases with UML, as discussed in [6]. Although those studies focused on transformations, and therefore are suited to be used with the MDD approach, their main concern was with the representation of the object oriented database structure with UML, without solving the mapping between relational databases and object oriented software.

The ORM tackles the impedance mismatch problem at implementation level, allowing the developer to use the relational and object oriented techniques together, abstracting the database structure [1]. The mappings can be used to both tell the framework how to translate objects to database tuples, as to generate a database structure [4].

The Java platform has a generic specification for ORM, the JPA persistence [4]. It defines guidelines to ORM frameworks and ORM software development behaviors, such as annotations, standard configuration files and rules for correct JPA coding. The JPA specification is adopted by several tools from major vendors (including at least one tool that works with *Microsoft .net*) and has rapidly become influential in the ORM setting as the most important, if not currently the only, specification to ORM.

MDD tools that offer UML extensions for persistence demands the adoption of a specific platform, imposing the modeling under certain design patterns and technologies, such as Data Transfer Objects [15]. The MD-JPA profile differs from those tools because it is specific to the ORM-JPA platform, but otherwise is a platform independent proposal.

This paper relies on the ATL language [8] to build transformations between models and in the the Java Abstract Syntax (JAS) meta model [7]. The JAS provides a complete mapping of the Java language to XMI meta model, allowing the transformation of Java code into models, as shown in [12]. The main advantage of ATL over the current open source QVT tools is that it allows both imperative and relational rules in the same transformation.

### 3. The MD-JPA Profile

The MD-JPA profile is mainly comprised of stereotypes, each of them extending a meta-class of UML, such as classes, properties or relationships. Besides the stereotypes,

constraints were defined in the Object Constraint Language (OCL) [16] to check if the models are well-formed according to the specification rules.

One of the key features of JPA is that it allows the writing of code with few details about software persistence and it allows a very detailed mapping that can be later used to create the database objects. The MD-JPA has this same characteristic, it has a set of detailed mappings that may be employed user, or he may just use a small set of stereotypes to design a persistent model.

Moreover, the MD-JPA as an UML profile takes advantage of the UML ability to design several class diagrams from the same model. It allows the use of diagrams with a higher level of abstraction (with a low number of persistence details) and diagrams with a complete view of the system, all over the same model. The following subsections specify the most important stereotypes of our JPA profile.

#### 3.1 Entities and properties

The entity concept of JPA is represented by the *Entity* stereotype suitable to UML classes. An entity denotes a class in which each instance is persistent and has a unique identifier. An entity may have a set of persistent properties that should have types according to the JPA mapping limitations. Unlike other approaches for persistence (see for example [1]), this profile does not distinguish persistent diagrams, allowing to draw entities and common classes together, in any class diagram, at users discretion.

By default each property of the entity is persistent, except for those marked with the *Transient* stereotype – the same rule applies to all descendants of the *Persistent* stereotype. Each entity hierarchy must implement the same access method for persistent properties, namely field or method access [4]. Therefore, the access method is defined in the entity stereotype, and not in properties and methods.

On JPA it is expected that each property presents *getter* and *setter* methods. To turn the models simpler, each UML property is considered as a *JavaBean* property [17] with private instance variable by default. No *getter* or *setter* method specification is required and the place of annotation is inferred by the access method of the entity.

Each persistent property may have a column mapping marked by the *Column* stereotype, enabling a more precise description of the type mapping to a database column. The *ColumnDefinition* object details column name, precision, scale, and if it allows null values, among other mapping details.

Entities may have one primary table definition and any number of secondary tables definitions, so that they can be used to define details about the database schema mapping and generation: name, catalog, schema, and unique constraints to name a few. For entities represented across more than one table, it is also possible to define the table of a property in the *ColumnDefinition* stereotype.

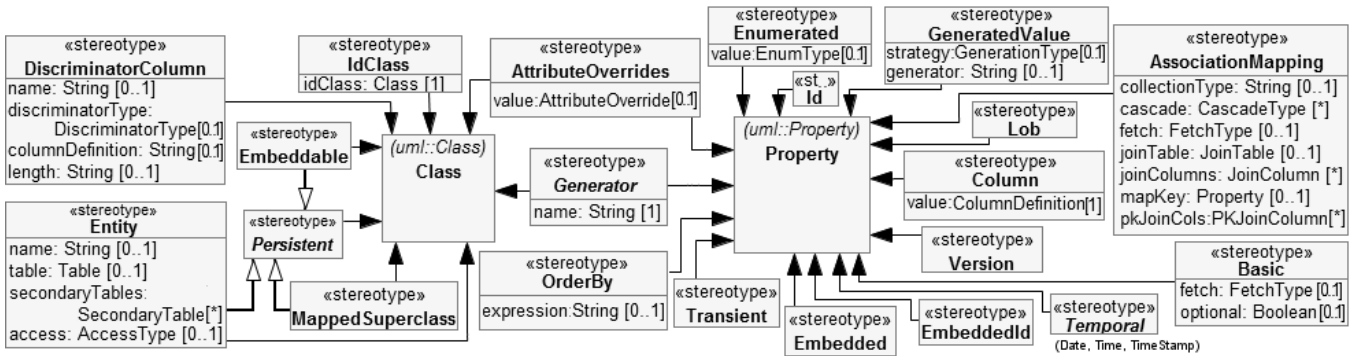


Figure 1. Main UML stereotype diagram of MD-JPA.

Other stereotypes can be applied on persistent properties, as shown in Fig. 1. The specializations of *Temporal* stereotype (*Date*, *Time* and *TimeStamp*) help mapping the Java *Date* class to a specific database type. The *Lob* stereotype marks a property as a large text or binary object. The *Basic* stereotype allows definition of database *Fetch* strategy and if the property allows null values in program level. The *Embedded* and *AssociationMapping* stereotypes will be explained in the further subsections.

### 3.2 Embeddable classes

Embeddable classes can represent part of the persistent state of entities. They do not have persistent identity, and each instance cannot belong to more than one entity instance at the same time [4]. Embeddable classes may be used to encapsulate common logic of a fine-grained component reused both in transient and persistent classes.



Figure 2. Entities and embeddable classes example.

The *Embeddable* stereotype indicates a class that can be embedded on entities. Each of its properties can have the *Column* stereotype describing how it might be mapped to database columns. The *Embedded* stereotype should be used with persistent properties that make reference to embeddable classes.

Fig. 2 is a diagram exhibiting a number of classes from an academic evaluation system, including entities and embeddable classes. The *DateInterval* class extends the transient *Interval* class by implementing an interval of time with day granularity. Each *Evaluation* instance stores a score for a predefined date interval, which is in turn an embedded object. The *CourseEvaluation* entity implements the abstract evaluation class and inherits the interval property, which is mapped to two date columns in the course evaluation table: *start* and *end*.

### 3.3 Primary Keys

Every entity hierarchy must have a primary key, defined in the topmost entity of the hierarchy tree [4]. Simple primary keys are just represented as annotated attributes. Composite

keys must have a separate embedded class, where each property is a column of the key. In such case, it is possible to specify one property that makes references to the embedded class or a set of properties that makes references to each of the embedded classes' properties.

There are four important stereotypes used on the representation of primary keys: (i) *Id* - Used to identify a single key, or each property of a composite key. Composite keys require the identification of the class with *IdClass*; (ii) *IdClass* - Identifies what class will be used to represent the primary composite key mapped by the *Id* stereotype; (iii) *EmbeddedId* - Defines a composite primary key as an embedded property; (iv) *GeneratedValue* - Defines a property as having automatic generated values. It is possible to define the generation strategy or to leave the decision to the ORM framework. It is also possible to describe a *TableGenerator* or a *SequenceGenerator*, both derived from the abstract *Generator* stereotype.

### 3.4 Relationships

All relationships between entities are, by default, persistent. The *mapped by property*, *mapping type*, and *direction* are respectively expressed on the model by the association relation, cardinality and navigability. The *one-to-one*, *many-to-one*, *one-to-many* or *many-to-many* mappings do not need any specific stereotype. Any cardinality above one represents a many relationship. For the sake of simplicity, associations are mapped to Java collections and Generics [14] is used to assure type checking. Aggregation and composition follow the same rules as associations.

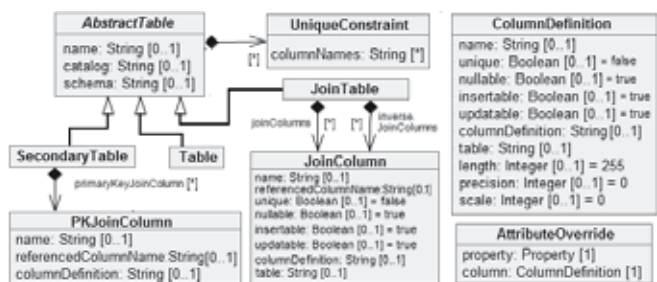


Figure 3. Table related classes in MD-JPA.

The stereotype *AssociationMapping* provides further definitions that allows to specify what kind of collection will be used (sets or lists), ordering expression, cascade type, fetch

strategy, and join tables. It is applicable on the *Property* element instead of the association itself: each UML class association has one property member in each side, and can be naturally mapped as connecting properties on the Java classes.

The join columns property of *AssociationMapping* can contain one or more mappings detailing how the foreign key should be generated. If the foreign key is part of the primary key, the *pkJoinCols* should be used instead. Another resource of JPA is the join table, defined as another property of *AssociationMapping* stereotype. It can define the structure of the table used to many to many or one to many unidirectional associations. Join tables, secondary tables, and the table definition are all meta-classes sharing the *AbstractTable* super meta-class, as shown in Fig. 3.

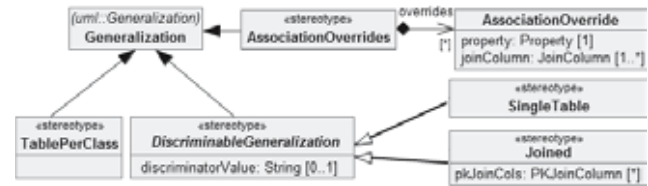
**Table 1. Annotations of JPA and equivalent stereotypes in MD-JPA.**

JPA	Stereotypes	UML Element
@Entity	Entity	Class
@Inheritance (Strategy)	SingleTable, Joined, TablePerClass	Generalization
@(One/Many) To (One/Many)	AssociationMapping	Property
@Embeddable	Embeddable	Class
@Embedded	Embedded	Property
@Transient	Transient	Property
@Id	Id	Property
@IdClass	IdClass	Class
@EmbeddedId	EmbeddedId	Property
@Column	Column	Property
@Version	Version	Property
@Enumerated	Enumerated	Class
@MappedSuperclass	MappedSuperclass	Class
@GeneratedValue	GeneratedValue	Property
@Lob	Lob	Property
@Temporal	Date, Time, TimeStamp	Property
@AttributeOverride(s)	AttributeOverrides	Property/Class
@OrderBy	OrderBy	Property
@DiscriminatorColumn	DiscriminatorColumn	Class
@SequenceGenerator	SequenceGenerator	Property/Class
@TableGenerator	TableGenerator	Property/Class
@AssociationOverride	AssociationOverrides	Generalization

### 3.5 Inheritance

An entity can generalize or specialize another class [4]. When the superclass is another entity, there are three possible strategies to object-relational mapping on JPA. For each, there is a stereotype that extends the *Generalization* element of UML, as shown on Fig. 4. If no stereotype is as-

signed, the inheritance strategy will be the default JPA strategy.



**Figure 4. Inheritance stereotypes of MD-JPA.**

The *SingleTable* stereotype represents a mapping of the entire hierarchy of classes to one table only. The *TablePerClass* stereotype represents a mapping where each concrete class has a separate table. In a similar way, the *Joined* stereotype also employs one table for each class, but the properties from the superclass are stored on the superclass table.

The discriminator column is mandatory to *SingleTable* but optional to *Joined* strategy, hence the *DiscriminableGeneralization* abstract stereotype can hold the discriminator value. The *DiscriminatorColumn* stereotype can be used to detail the column – or all those information can be left to the framework decision. The *Joined* can also be detailed with foreign key to primary key mappings.

Table 1 summarizes the annotations and their equivalent stereotypes on this profile. The following classes were defined to complement those stereotypes with tagged values: *JoinColumn*, *ColumnDefinition*, *Table*, *SecondaryTable*, *JoinTable*, *AttributeOverride*, *UniqueConstraint*, *DiscriminatorValue*, *PKJoinColumn*, and *AssociationOverride*.

### 4. Constraints for model checking

One of the advantages of using extensions such as MD-JPA over standard class models is the ability to check the model validness against the specific modeled domain. Not all models can be mapped by the JPA, there are limitations about data type use, inheritance and relationship. This can lead to discrepancies between pure class models and their implementation that often may never be corrected in the original model.

In order to check models with the persistence profile, we specify and implement a set of constraints using the Object Constraint Language (OCL) [16]. Stereotypes and UML meta-classes can have a set of invariant constraints that must assert true for all its elements, detecting which elements are bad formed.

The JPA specification states several requirements for entities, fields, keys, inheritance, and relationships; this specification was the base to create most of our OCL invariants. For those invariants, the name correspond to the chapter, section, and paragraph of the specification that is under check, hence *rule21\_p3* references chapter 2, section 1 paragraph 3. When it states that “The entity class must be a top-level class”, the constraint needs to check if the base class of the entity is not a nested class [14], what in UML is expressed by the owner relationship [11].

```

inv rule21_p3 : (base_Class.owner.oclIs-
KindOf(uml::Package) or base_Class.own-
er.oclIsKindOf(uml::Model))

```

The context of *rule21\_3* is the abstract *Persistent* stereotype, which generalizes *Entity*, *MappedSuperClass* and *Embedded* stereotypes. It provides a common *base\_Class* variable pointing to the class element and its opposite variable *extension\_Persistent* was redefined by each sub stereotype.

A class whose owner is another class or anything other than a package of model cannot be considered as top level class [14]. Besides this, a top level class cannot be protected or private, as shown in the following continuation of the rule:

```

and not (base_Class.visibility=uml::Vis-
ibilityKind::protected or
base_Class.visibility=uml::Visibility-
Kind::private)

```

The profile has a set of another 30 invariant checks for model well-formedness, not detailed here due to space limitations. About half the rules were direct references to the specification of JPA, and the remaining rules are additional checks to enforce the good use of the profile and UML modeling for JPA. Some examples of additional checks are: only one class can generalize any other class in Java; the *AssociationMapping* stereotype can only be applied to the association ends; if a property is overridden, it must exist in one of the superclasses.

## 5. Transformations

The present section validates MD-JPA profile as a MDD artifact by showing the ways through which it is used as a development artifact. It first proposes a set of ATL transformations rules that takes MD-JPA models, generating models of Java annotated implementation - Java models will be translated into code, and most ORM frameworks can generate the database structure from this code [4]. To further check the transformation completeness, a model with the example implementation of JPA specification was created by reverse engineering, comparing the results with the example code.

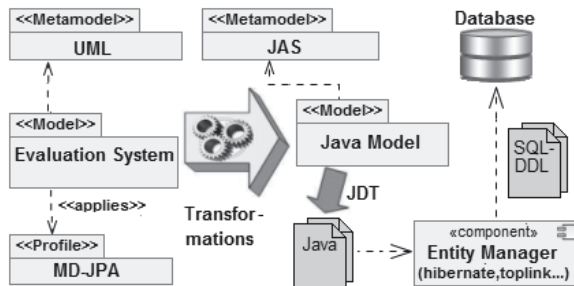


Figure 5. Transformation Overview.

This target model will later be translated into code. Fig. 5 shows an overview of our transformation process. Our ATL transformation is a set of rules that match elements in a source model, following the source meta model, and pro-

duces new elements in the target model, according to the Java abstract syntax tree meta model defined by [7].

Classes, properties, methods, enumerations, and interfaces are matched by the rules, all among UML base elements. Each matched element produces one or more Java elements: A top level class will generate a class and a program unit; a property will generate a private instance variable and accessors methods.

Some elements cannot be produced solely by match rules and are addressed by imperative rules called inside some match. For instance, the modifiers generated from a single class, attribute or operation, depends on the visibility and the abstract attribute. In this case there is not a clear source for the generated element other than the class or feature, and it is generated only in certain circumstances – a private visibility or abstract being true. The imperative rules are an alternative to create one match rule for each combination of circumstances in the source element [8].

The resulting model will be a model in the JAS meta model, which is based upon the eclipse JDT framework [5]. The transformation of the JAS model into code is performed with the implementation of a JDT visitor.

In order to validate the transformations we present a model created from the JPA example system, comparing the generated code with the implementation proposed at the JPA specification [4]. Fig. 6 shows a detailed MD-JPA class diagram with each class, property, and applied stereotype containing the mapping information of the *acme* example.

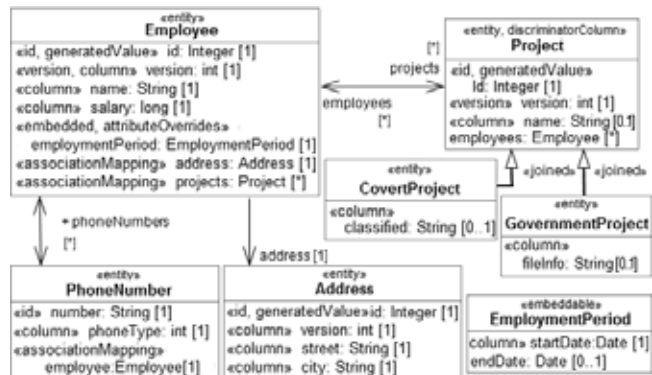


Figure 6. “acme complex example” of JPA specification.

The only semantic differences between the two “acme” sources were the instance variable names, when the property name is distinct from the private variable, and the use of generics in all relationships. Both are due to decisions concerning the mapping of Java concepts of properties and collections in MD-JPA profile. However, the database generated by hand written and generated code was identical, both with Open-JPA [20] and Hibernate [21] JPA implementations, for all available database dialects on those tools.

In order to tie together the MD-JPA profile and transformations, an open source eclipse “plugin” was developed, although they can be used outside eclipse platform [23].

More details about the transformation rules, not included by space restrictions, can be found at the MD-JPA website [22].

## 6. Conclusions

Despite the wide adoption of UML, there is not a clear, consensual and easy-to-use solution for the problem of persistence modeling. Separate ER and UML models are possibly the most common approach. However, the MDD approach demands integrated models that can correctly represent the implementation of the system.

This paper has presented a UML profile which allows the modeling of the persistent elements of a system, following the JPA persistence standard. The focus of MD-JPA profile is the creation of models depicting jointly both persistent and object oriented elements. Those models can be checked, in order to detect a design incompatible with ORM, and then transformed in parts of the software implementation. Moreover, it allows the use of ORM advanced resources as needed, like the code only approach; and preserves the UML ability to create distinct diagrams with different levels of abstraction, hiding the unnecessary details in top level diagrams, without losing the mapping information stored in the low level diagrams.

A simple example illustrates how we can validate the MD-JPA profile by the use of transformations, and also shows the utility of our approach: a model was built according to the complete actual example available in the JPA specification. This model was then transformed using our defined ATL transformations into an implementation that is equivalent to the example, thus showing that in fact our MD-JPA profile can represent adequately the annotation mappings of JPA.

Future studies might extend this profile to attend constructs of specific ORM tools. A study of all common features among ORM tools can lead to a canonical profile, that can later be integrated as higher level profile of MD-JPA.

## 7. REFERENCES

[1] Ambler, S. W.: *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. Wiley Publishing, Inc, USA (2003)

[2] Beydeda, S., Book, M., Gruhn, V.: *Model Driven Software Development*. Springer-Verlag New York (2005)

[3] Chen, P. P-S.: The Entity-Relationship Model-Toward a Unified View of Data. *ACM TODS Vol 1, No 1* (1976)

[4] Demichiel, L., Keith, M.: *Enterprise JavaBeans™, Version 3.0, Java Persistence API* <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>

[5] Eclipse Java Development Tools (JDT), <http://www.eclipse.org/jdt>

[6] Grant, E. S., Chennamaneni, R., Reza, H.: Towards analyzing UML class diagram models to object-rela-

tional database systems transformations. *Proceedings of the 24th IASTED international conference on Database and applications*, pp. 129--134, Innsbruck, Austria (2006)

[7] Jas-metamodel, <http://www.eclipse.org/gmt/modisco/toolBox/JavaAbstractSyntax/>

[8] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., Valduriez, P.: *ATL: a QVT-like Transformation Language*. *OOPSLA'06*, pp. 719--720, Portland, Oregon, USA (2006)

[9] Mellor S. J., Scott K., Uhl A., Weise D.: *MDA Distilled: Principles of Model-Driven Architecture*. Addison Wesley, USA (2004)

[10] OMG: *Request For Proposal: Information Management Metamodel*, <http://www.omgwiki.org/imm/doku.php>

[11] OMG: *Unified Modeling Language: Superstructure*, <http://www.omg.org/cgi-bin/doc?formal/07-02-03>

[12] Pires, W., Brunet, J., Ramalho, F.: *UML-based design test generation*. *Proceedings of the ACM symposium on Applied computing*, pp. 735--740, Fortaleza, Ceara, Brazil (2008)

[13] Rumbaugh, J., Jacobson, I., Booch, G.: *The Unified Modeling Language Reference Manual*. Addison-Wesley Longman Ltd (1999)

[14] *Java Language Specification*, <http://java.sun.com/docs/books/jls/download/langspec-3.0.pdf>

[15] Caliari, G., Muniz Silva, P.: *A case study on modeling persistence with MDA tools*. *Anales de XII Jornadas de Ingeniería del Software y Bases de Datos*, pp. 51-59, Zaragoza (2007)

[16] OMG: *Object Constraint Language*, <http://www.omg.org/docs/formal/06-05-01.pdf> (2006)

[17] *JavaBeans specification*, <http://java.sun.com/javase/technologies/desktop/java-beans/docs/spec.html>

[18] Song, E., Yin, S., and Ray, I.: *Using UML to model relational database operations*. *Comput. Stand. Interfaces* 29, 3, pp. 343--354. DOI=<http://dx.doi.org/10.1016/j.csi.2006.05.006> (2007)

[19] Luján-Mora, S., Trujillo, J., and Song, I.: *A UML profile for multidimensional modeling in data warehouses*. *Data Knowl. Eng.* 59, 3, pp. 725--769. DOI=<http://dx.doi.org/10.1016/j.datak.2005.11.004> (2006)

[20] *Apache OpenJPA*, <http://openjpa.apache.org/>

[21] *Hibernate*, <http://www.hibernate.org>

[22] *MD-JPA plugin*, <http://wiki.inf.ufrgs.br/mediawiki/index.php/MD-JPA>

[23] *Eclipse platform*, <http://www.eclipse.org>

# A Pragmatic UML-based Meta Model for Object-oriented Code Generation

Tobias Haubold, Georg Beier, Wolfgang Golubski

Zwickau University of Applied Sciences, Informatics, Zwickau, Germany

E-mail: tobias.haubold@fh-zwickau.de

## Abstract

*Model-based or model-driven software development is a highly regarded topic as it claims to development high quality software faster. It is already used in the software industry. There are efforts to use UML as well as DSLs as meta model but either they lead to complex model transformations in case of UML or the model transformations are restricted in their reuse in case of DSLs. We target this problem by introducing a fixed meta model for code generation resulting in a fixed, reusable back-end of an MDSD process. This approach overcomes the use of the modeling meta model in model transformations resulting in a higher return of invest in those and the whole MDSD process.*

## 1. Introduction

The increasing number of acronyms, e.g. Model-Driven Software Development (MDSD), Model Driven Architecture (MDA), Model Driven Engineering (MDE) or Model Driven Test Development (MDTD) to classify model driven approaches reflect the interest in this research topic and are already used in the software industry [1, 2, 3]. The key concept is to automatically derive source code from deterministic evaluable models. These models use prescribed structures and semantics which are defined within meta models.

This involves the Unified Modeling Language (UML)[4] as a standard to describe object oriented software systems which is specified by the Object Management Group (OMG)[5]. It plays a key role in the Model Driven Architecture (MDA)[6]. It evolves with an increasing amount of supported charts and modeling concepts to a quite complex language[7]. The UML meta model as the data model is consequently normalized which further increases its complexities and outlines the ineligibility to access a particular model information. The *Executable UML* uses a subset of the UML namely class models and state machines with an additional action language [8]. Activity charts are an important aspect in model driven testing [9] to specify flows. Both examples underline that only a subset of the thirteen diagrams[4] of the UML is relevant for model driven ap-

proaches.

Today Domain Specific Languages (DSL) are very popular. They focus on the concepts of a particular domain. The foundation of DSLs is the domain concept of Shlaer and Mellor[10] and therefore DSLs exist for several abstraction levels in software development. DSLs are normally more handy and precise but only applicable for a certain field of application or technical aspect. One classification of DSLs is into subject area DSLs and system aspect DSLs [11]. DSLs can be defined using UML profiles or using other meta modeling languages like MOF[12] or Ecore[13, 14].

Both the UML and DSLs play a key role in model driven approaches by providing a meta model (there can be more) for the MDSD process, but have also a wide field of application out of the scope of model driven approaches. The decision to choose meta models is not an either or - UML and DSLs can be used together. But the choice of meta models has an impact on the model transformations as they rely on these meta models. Thus changes on meta models cause changes of model transformations.

Model to text transformations generate source code for a particular programming language. These mapping functions usually use the archetype[15] or template[16] concept to accommodate the complex abstract syntax (the meta model) of programming languages but they remain considerably. Based on these facts, the following consequences can be drawn:

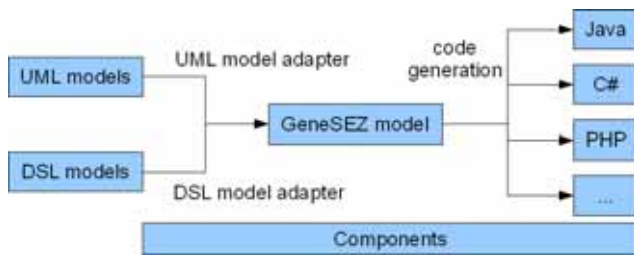
- the use of UML leads to complex transformations
- the use of subject area DSLs increase the gap between the abstractions in programming and modeling languages resulting in more complex transformations
- system aspect DSLs are restricted in their use to the technical aspect they cover
- domain changes in subject area DSLs have more serious consequences that changes in system aspect DSLs
- if a DSL cannot be applied another DSL has to be invented resulting in the creation of transformations
- if a subject area DSL cannot be applied and there is no intermediate meta model in the MDSD process, the whole MDSD process has to be reinvented

The GeneSEZ approach as outlined in section 2 overcomes

these weaknesses by introducing a separate meta model for the MDSD process which is covered in section 3. With an intermediate transformation step the modeling meta model is decoupled from the code generation and the GeneSEZ meta model makes the development, maintenance and customization of model transformations easier as outlined in 4. We discuss related work in section 6 and conclude our approach in section 7.

## 2. The GeneSEZ MDSD Approach

As illustrated in Fig. 1 our approach consists of four main concepts:



**Figure 1. Overview of the GeneSEZ MDSD process**

**Model adapters** are used to populate GeneSEZ models with information

**GeneSEZ meta model** is the meta model of our MDSD process

**Components** are several utilities supporting the MDSD process, e.g. execution of model transformations

**Platform projects** support code generation for programming languages

With the use of model adapters our approach does not prescribe a particular meta model to model the application. In fact, we believe this decision should be based on other means than the MDSD process and therefore we decoupled the meta models used for modeling and code generation.

The GeneSEZ meta model is the fixed meta model within our MDSD process enabling reusable model transformations across different application meta models. Model transformations enabling code generation for different target programming languages as well as frameworks and libraries.

## 3. The GeneSEZ Meta Model

The GeneSEZ meta model is a general purpose meta model for object oriented software systems and can be seen as a domain specific meta model where the domain[10] is

object oriented source code generation. It targets the need of developers for a special view of the information of an application model with respect to the creation and maintenance of model transformations. We target especially model to text transformations to achieve a close mapping between the meta model and the source code. Therefore it has some commonalities with UML class models.

### 3.1. Object oriented constructs

To support object oriented programming languages the constructs shown in figure 2 are used to describe the static structure of an application. The following constructs need to be distinct from the UML:

- the *final* attribute of an *MClassifier* specifies if a classifier can be inherited or not
- the *final* attribute of an *MOperation* specifies if it can be overridden or not
- the *final* attribute of a *MProperty* specifies if its value is changeable or read-only
- the concept of association ends
- the distinction between attributes and association ends

The UML supports associations with more than two ends[4]. To accommodate the complexities of mapping associations into source code as outlined in [17] the GeneSEZ meta model currently supports only binary associations<sup>1</sup>. To indicate this difference we use the name *association role* instead of *association end*.

Attributes and association ends are both properties in the UML[4]. The distinction in UML takes place if a property has a reference to an association or not, i.e. the check of an attribute value. The GeneSEZ meta model introduces a separate model element for association roles. Distinct model elements have the benefit that properties and association roles can filtered by their type. Therefore the distinction is done on the meta model layer within a GeneSEZ model instead of the application model level within an UML model (i.e. based on an instance of the UML meta model).

### 3.2. Type system

The GeneSEZ type system consists of the types shown in figure 3. All types inherited from *MClassifier* are defined within the target programming language. The concept of generic and external types are not known by the UML:

- parameterized types are specified with UML templates
- the UML has no support to use already existing types

Parameterized types are supported using the concept of generic types. A generic type (*MGeneric*) is a place holder

<sup>1</sup>furthermore we are not aware of the need of and a mapping for more than two association ends

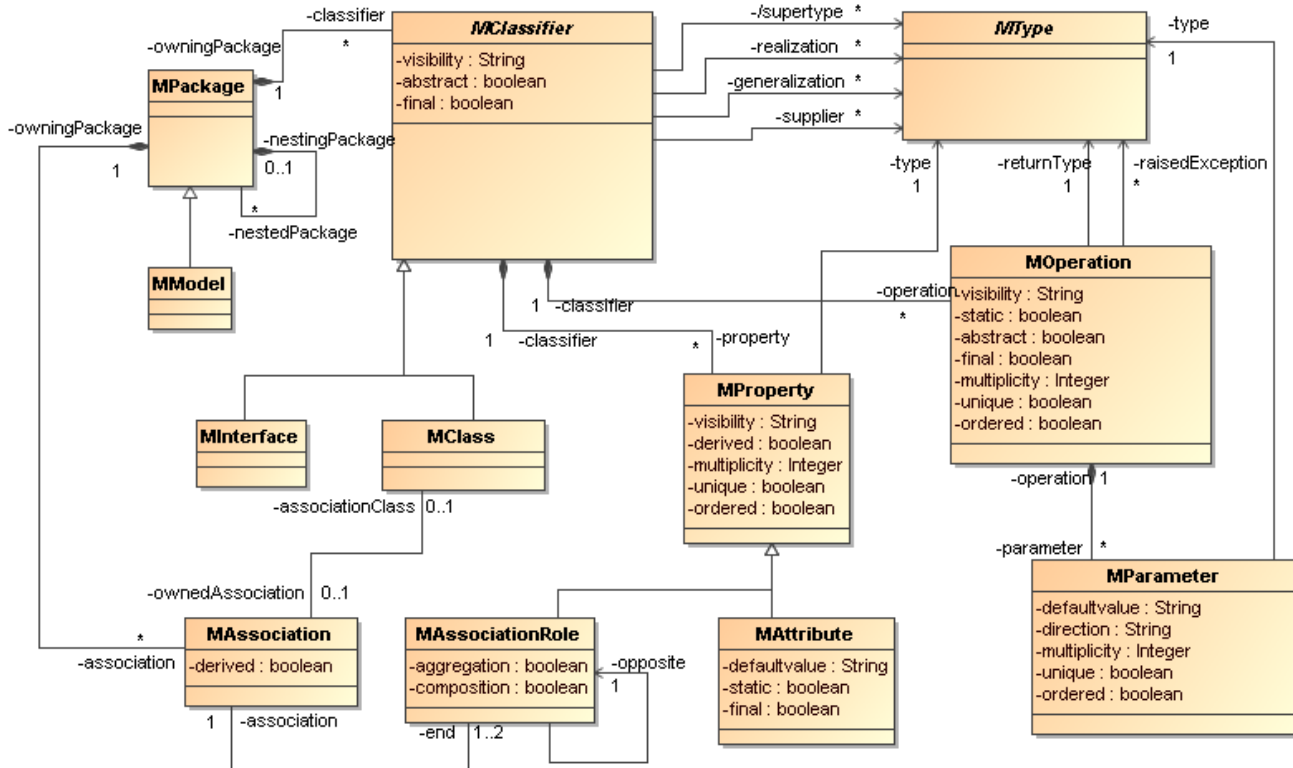


Figure 2. The definition of object oriented constructs in the GeneSEZ meta model

for a type of the programming language and is defined as a textual specification. This definition overcomes the complex evaluation of formal and actual type parameters within UML templates. When using parameterized types in UML formal template parameters have to be substituted with actual template parameters. In the case of code generation for a target programming language supporting parameterized types it is only of interest which textual representation has to be generated. It does not matter if it is a formal template parameter or an substituted one.

To support the reuse of existing types in programming languages, frameworks or libraries we introduce the concept of external types (*MExternal*). This is of particular interest of software developers as they always prevent to reinvent the wheel. External types are not possible with the UML and results in filter expressions every time the definition of an UML classifier has to be generated and therefore for every target programming language. With the GeneSEZ meta model this filter expression is not needed resulting in a simpler and more understandable mappings to target programming languages.

Primitive and external types are mapped to already existing and available types in a programming language.

### 3.3. Special support for code generation

The term *special* is used to classify meta model definitions which are either not needed to run the application or are not directly contained within the generated source code. This involves the following three concepts shown in fig. 4:

**Comments** can be used to annotate model elements with textual content, usually used to generate the source code documentation.

**Stereotypes** are an annotation mechanism for model elements and usually used as marker to control and adjust model transformations.

**Tagged Values** are a concept to specify additional information for model elements. The additional information can be used to embed it into the source code or as a kind of parameter to control model transformations.

The UML uses profiles to extend the meta model in a lightweight way. Stereotypes are UML classifiers which extend an existing meta class. Tags are attributes of a stereotype and tagged values the attribute values. This concept is specifically designed to allow the definition of DSLs based on UML. However, for code generation other semantics are needed which we cover with *MStereotype*, *MTag* and



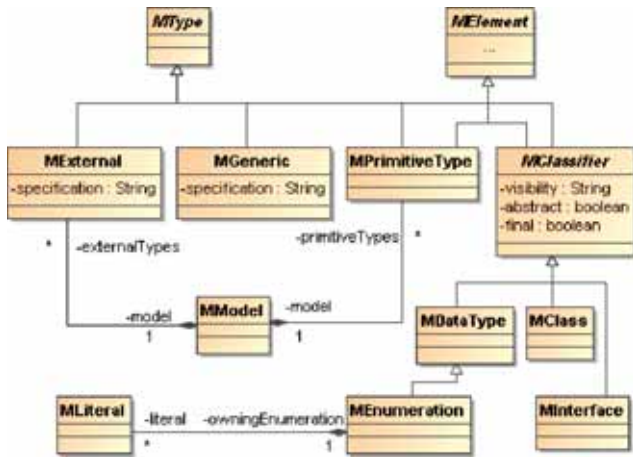


Figure 3. The definition of types in the GeneSEZ meta model

*MTaggedValue*. This involves the use of stereotypes as annotations rather than specialized types as further outlined in section 6.

#### 4. Sample Application

With the development of an application for project planning we will illustrate our approach. The domain model of the application is shown in figure 5. It allows to structure a project into milestones and tasks. Tasks can be nested using main tasks. Each task has a planned time effort. The time effort of several work units of a sub task can be booked by each person working on that task. This time effort is called time budget and modelled as an association class between *SubTask* and *Person*. Main tasks have a derived attribute which indicates the current time spent on this task.

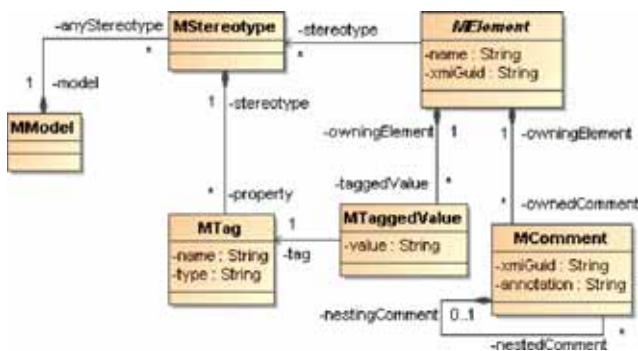


Figure 4. The definition of stereotypes, tagged values and comments in the GeneSEZ meta model

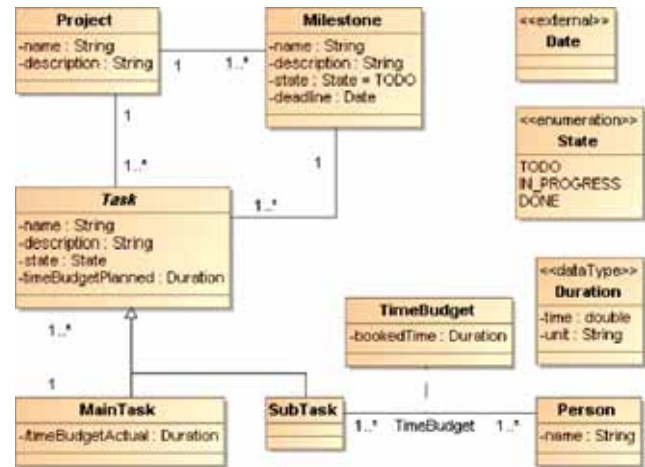


Figure 5. The domain model of the sample application *time budget planning*

To define external types with the UML we defined the stereotype *external* and assigned it to the class *Date*. This supports expressive and robust models. During the transformation into a GeneSEZ model every classifier with this stereotype is transformed into an *MExternal* instead of an classifier. This results in a clear mapping that all classifiers are defined within the source code. The filter is only specific to the UML to GeneSEZ transformation instead of the UML to target programming language transformation. During a GeneSEZ to target programming language transformation such a filter does not need to be applied.

Distinct model elements for properties and association roles have the benefit that properties and association roles can be distinguished on the meta model layer instead of the application model layer. Application model layer based distinctions involve additional checks of the existence of the referenced objects by checking the attribute value.

#### 5. Evaluation

Our primary focus was to ease code generation using UML models by introducing the GeneSEZ meta model as an intermediate transformation step before code generation in the whole MDSD process. The GeneSEZ model is created by a representing mapping[15] of a UML model. This has several benefits:

- a handy and concise general purpose meta model for model transformations
- the whole meta model can be printed on two pages in letter format for an handy overview during model transformation development rather than searching in bloated UML API documentation

- increased productivity during development, maintenance and customization of model transformations
- a fixed and reusable back-end for code generation within an MDS process which is decoupled from the meta model used for modeling
- UML as well as subject area DSLs can be used to populate GeneSEZ models resulting in robust models which concentrate on the application domain
- stereotypes and tagged values can be used to provide annotations and additional information for model transformations, e.g. to support system aspect DSLs
- model transformations become a reusable and customizable set of architectural building blocks across different modeling meta models

The GeneSEZ meta model evolved during the application in projects of our industry partners to a practical approach. The field of application includes:

- code generation for embedded systems based on Java (JavaME)
- development of Java applications
- reengineering and porting of a software written in C++ to C#
- development of PHP based web applications
- development of Java applications based on EJB3 and the Seam framework

In these projects we were able to generate 50% - 80% of the source code which indicates the importance of the structural part of an application.

## 6. Related Work

The UML as well as DSLs are used in model driven approaches which can be classified by:

- the use of the UML meta model
- the use of the DSLs which were used to model the application
- the use of special DSLs which are populated by the information of UML models

There are two popular open source tools for model driven approaches which we will cover in more detail: *openArchitectureWare*[18] and *AndroMDA*[19].

Commercial tools focus mainly on DSLs, e.g. *MetaCase*[20] or *Jetbrains MPS*[21]. The latter one is a language oriented tool that supports the modularization and reusability of already defined languages. Languages can be created, embedded into other languages and combined. Furthermore an application is created by structured editing which directly edits the syntax tree.

### 6.1. AndroMDA

AndroMDA is currently an UML-based approach and uses an UML implementation in Java as meta model for code generation. Tool adapters are used to support a couple of UML tool vendors. For meta model adjustments and information hiding AndroMDA has the concept of so called *meta facades*: Java interfaces and classes according the *facade* pattern<sup>2</sup> to provide a special view of the information in the UML model to a cartridge<sup>3</sup>. Usually every cartridge has its own meta facades. The meta facades use either the UML meta model directly or other meta facades. In fact, the implementation of meta facades can be seen as model to model transformations implemented in Java.

Developers who create, maintain or customize cartridges have to work with the complex UML meta model. Currently all model transformation logic has to be implemented using Java. This is sometimes a bit intricately, e.g. to determine qualified names of a model elements. For such calculations a functional expression language like *openArchitectureWares Xtend*[23] is more suited because it is more expressive for this kind of problems.

### 6.2. openArchitectureWare

The generator framework *openArchitectureWare* supports various meta models:

- the EMF UML2 implementation and the so called *classic* meta model for UML based approaches
- EMF and Java beans -based meta models, primarily used for DSL based approaches
- external textual DSLs based on *Ecore* with *Xtext*[24]

The so called classic meta model is a simplified version of the UML meta model. To use stereotypes and tagged values the meta model classes must be extended by one implementation class for each stereotype. Tagged values are the attributes of the stereotype classes. Due to the importance and heavy usage of stereotypes and tagged values in code generation this is a tedious additional work. This concept treats stereotyped classes as special types, which is closer to the UML semantics but leads to challenges during model transformations if multiple stereotypes are assigned to one model element. These challenges are avoided using the GeneSEZ meta model because we treat stereotypes as annotations (see section 3.3). The classic meta model seems to be developed in a mixed way by using MDS and manual implementations in contrast to the completely generated GeneSEZ meta model. The future of the classic meta model was a long time unclear but it seems to be further supported.

<sup>2</sup>The facade pattern provides a simplified interface to a more complex implementation consisting of at least one class [22]

<sup>3</sup>AndroMDA organizes model transformations in cartridges according the target programming language and supported frameworks.

### 6.3. Sculptor

A DSL based approach for code generation is *Sculptor*<sup>4</sup>. It can be used to generate web applications based on Spring, Spring Web Flow, Hibernate, Java EE and Java Server Faces. It is build on top of openArchitectureWare and uses Xtext to describe the DSL and to generate an editor for the DSL. The DSL is then used to describe the web application. This approach is well suited for this specifically tailored domain. But the meta model as well as the model transformations can only be used for this kind of applications. If the same application should be ported to a desktop application, the model cannot be reused because it is build with the concepts of web applications.

## 7. Conclusion and further work

There was a big effort to establish the UML as a standard notation to describe object oriented software. The use of standards is always preferable to support the elaboration of communicable and understandable models. Unfortunately MDSO and UML evolved to digressing technologies [25]. The MDA standard comes with good concepts but cannot hide the complexities of the UML [7] and its ineligibly for code generation due to the huge information contained.

Our approach has still some shortcomings. The meta model still covers only the structural aspects of an application and the concept of generic types needs some attention. It also contains only navigable association roles which is a discussion point. By now, return types of operations cannot have a comment.

Further work has to be done to support standardized stereotypes. Current work includes meta model support for state machines, C++ as a target platform and the enhancement of the existing target platforms with common and popular frameworks. Last but not least we work on a platform independent concept to implement associations. All work is done within projects of our industry partners.

Moreover a new project is initiated to invent the benefits of expressing robustness, testability and security in UML models utilizing the GeneSEZ approach to develop automotive software applications. The project will be carried out with a leading German automobile company as an industry partner.

## References

- [1] Use of openarchitectureware in the software industry. <http://www.openarchitectureware.org/index.php?topic=success>.
- [2] Use of andromda in software projects. [http://galaxy.andromda.org/index.php?option=com\\_content&task=blogcategory&id=26&Itemid=40](http://galaxy.andromda.org/index.php?option=com_content&task=blogcategory&id=26&Itemid=40).
- [3] Use of metacase in software projects. <http://www.metacase.com/cases/>.
- [4] Object Management Group. *The UML Superstructure Specification 2.1.2*, November 2007.
- [5] Homepage of the object management group (omg). <http://www.omg.org/>.
- [6] Object Management Group. *Overview of Model Driven Architecture (MDA) specifications*. <http://www.omg.org/mda/specs.htm>.
- [7] Dave Thomas. Uml - unified or universal modeling language? *Journal of Object Technology*, 2(1):7-12, January-February 2003.
- [8] Stephen J. Mellor and Marc Balcer. *Executable UML: A Foundation for Model-Driven Architectures*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. Foreword By-Jacobson,, Ivar.
- [9] Paul Baker, Zhen Ru Dai, Jens Grabowski, Ø ystein Haugen, Ina Schieferdecker, and Clay Williams. *Model-Driven Testing: Using the UML Testing Profile*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [10] Sally Shlaer and Stephen J. Mellor. The shlaer-mellor method. 1996.
- [11] Johan den Haan. Dsl and mde, necessary assets for model-driven approaches. [http://www.theenterprisearchitect.eu/archive/2008/08/11/dsl\\_and\\_mde\\_necessary\\_assets\\_f](http://www.theenterprisearchitect.eu/archive/2008/08/11/dsl_and_mde_necessary_assets_f), August 2008.
- [12] Object Management Group. *The Meta Object Facility (MOF) Core Specification 2.0*, Januar 2006. <http://www.omg.org/spec/MOF/2.0/>.
- [13] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. J. Grose. *Eclipse Modeling Framework - A Developer's Guide*. Pearson Education, 2004.
- [14] Homepage of the eclipse modeling framework (emf). <http://www.eclipse.org/modeling/emf/>.
- [15] Stephen J. Mellor, Scott Kendall, Axel Uhl, and Dirk Weise. *MDA Distilled*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.
- [16] Krysztof Czarnecki and Simon Helson. Classification of model transformation approaches. In *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*, 2003.
- [17] Gonzalo Gnova, Carlos Ruiz del Castillo, and Juan Llorens. Mapping uml associations into java code. *Journal of Object Technology*, 2(5):135-162, September-October 2003.
- [18] Homepage of openarchitectureware. <http://www.openarchitectureware.org/>.
- [19] Homepage of andromda. <http://andromda.org/>.
- [20] Homepage of metacase. <http://www.metacase.com/>.
- [21] Homepage of the jetbrains meta programming system (mps). <http://www.jetbrains.com/mps/>.
- [22] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.
- [23] *Reference of the Xtend language*. [http://openarchitectureware.org/pub/documentation/4.3.1/html/contents/core\\_reference.html](http://openarchitectureware.org/pub/documentation/4.3.1/html/contents/core_reference.html).
- [24] Homepage of the xtext language. <http://www.eclipse.org/modeling/tmf/?project=xtext>.
- [25] Sven Efftinge, Peter Friese, and Jan Köhnlein. Best practices for model-driven software development. *InfoQ*, June 2008. <http://www.infoq.com/articles/model-driven-dev-best-practices>.

<sup>4</sup>Sculptor is a DSL based approach to MDSO, [http://www.fornaxplatform.org/cp/display/fornax/Sculptor+\(CSC\)](http://www.fornaxplatform.org/cp/display/fornax/Sculptor+(CSC))

# An Ontology-based Model Driven Approach for a Music Learning System

Yingchun Tian, Feng Chen and Hongji Yang  
Software Technology Research Laboratory  
De Montfort University, Leicester, LE1 9BH, England  
{ytian00, fengchen and hyang}@dmu.ac.uk

Leigh Landy  
Music, Technology and Innovation Research Centre  
De Montfort University Leicester, LE1 9BH, England  
llandy@dmu.ac.uk

## Abstract

E-learning systems are attracting much attention in both research and industry areas. Because of massive data and complex functions, the means how to organise the information and build architecture effectively to improve the e-learning system development is considered to be a kernel issue. This paper proposes an ontology-based model driven approach for a music learning system named the Pedagogical ElectroAcoustic Resource Site (EARS II). Firstly, requirements are extracted into vocabularies under Natural Language Processing (NLP) theory. Then, a Reference Ontology (RO) is designed based on Learning Technology System Architecture (LTSA) and vocabularies are classified into Application Ontology (AO) based on RO. Finally, Platform Independent Model (PIM) is generated from AO following proposed transformation rules. Furthermore, the implementation of the proposed approach into EARS II is provided, showing it's feasible and applicable to facilitate the modelling process and increase the maintainability and reusability of the implemented system.

The proposed approach has the potential to develop general e-learning systems without the user is connoisseur of philosophy ontological or pedagogical, while giving him the approach steps that works in an Object-Oriented environment.

**Keywords:** *Ontology, Learning Technology System Architecture (LTSA), Model-Driven Approach, E-learning, ElectroAcoustic Resource Site II (EARS II), Electroacoustic Music*

## 1. Introduction

In recent decades, e-learning has been widespread, especially since standardising initiatives for learning technologies have begun. Therefore, system's quality is more concerned, not only in computer area but also

pedagogical domain. Specific background knowledge and pedagogical approach are necessary but easily ignored. Traditionally, team work is the solution that involves both software developers and pedagogical experts. However, the common problem is the team cannot start quickly because the huge gap between those two areas. Accordingly, this paper offers an approach as solution. Learning Technology System Architecture (LTSA), which contains educational knowledge, is combined with ontology to provide pedagogical support. Meanwhile, Model-Driven Architecture (MDA) has a significant idea, which is the independence of specification and particular platform. So Platform Independent Model (PIM) is considered befitting form of final architecture in this paper.

Besides, a real music learning system is involved. The musical corpus that will form the focus of this research is called Electroacoustic music, which is "any music in which electricity has had some involvement in sound registration and/or production other than that of simple microphone recording or amplification" [1]. This corpus is useful for a number of reasons not least because of the fact that those who make it are highly involved with technology. Therefore introducing its pedagogy online might be seen as a logical thing to do. The complete pedagogical environment will be called the ElectroAcoustic Resource Site II (EARS II) and is a follow-up to the internationally respected research resource site, EARS (<http://www.ears.dmu.ac.uk>) which has been supported by the Arts and Humanities Research Council in the UK as well as by UNESCO.

Above all, this paper proposes an ontology-based model driven approach for a music learning system. It will result in PIM, which contains pedagogical information and becomes a bridge for the areas' gap. The rest of the paper is organised as follows. Section 2 is a short introduction about related work; Section 3 provides the whole approach this paper proposes, including vocabularies extraction, ontologies classification, and transformation into PIMs; Next section is the implementation of this approach into the Pedagogical ElectroAcoustic Resource Site II (EARS II) [2] project. Finally, Section 6 concludes the paper with a summary of further work.

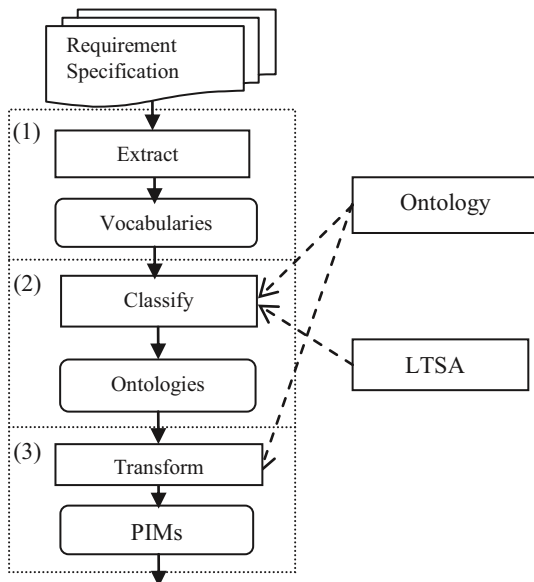
## 2. Related Work

This section discusses research related to our work. Gavras et al [3] have proposed an MDA-based development methodology. Applying MDA to enterprise computing have described in [4]. In [5] author has provided model driven software modernisation. They proved the practicability to apply MDA in general systems' development. Even in software evolution, MDA is an effective methodology [6]. However, those related works mentioned above only focus on MDA's application but not specific to PIM's establishment. Though Solms and Loubser [7] formulated a methodology to generate PIM, it aims at the system domain experts but not software technicians.

The initial design for the MDA-based development of EARS II is provided as a paper [8], including lifecycle and pedagogical design. In this paper, our proposal is based on it and can be considered as an extension and specification of the lifecycle.

## 3. The Proposed Approach

The proposed ontology-based model driven approach for music learning system is shown as Figure 1. Actually, this approach is not only limited on music-learning system but also suitable for general e-learning systems.



**Figure 1: Ontology-based PIM Modelling Approach**

There are mainly three steps in this approach:

- (1) Extracting vocabularies: according to Natural Language Processing (NLP) technology, requirements are extracted into vocabularies.

- (2) Classifying ontologies: LTSA is the basic structure for classify the vocabularies that come from previous step. First, a RO is involved in this phase, which designed based on LTSA. Then classify vocabularies into RO to be an AO. Next task is to add extra vocabularies into AO. Finally, if there are redundancies in AO, they are reduced in this step.
- (3) Transforming into PIMs: Ontologies are transformed into Platform Independent Models following a set of transformation rules that we proposed. Considered the PIMs are showed as a set of UML diagrams generally, following the five rules, classes are generated with name, mandatory attributes, operations, interfaces, and relationships.

### 3.1 Vocabularies Extraction

Vocabularies extraction is always happened as a general activity in initial development such as requirement writing. Normally, developers extract them on mind with potential self-rules. In this approach, Natural Language Processing (NLP) theory is used as basic technology for extraction. This activity aims to get simple vocabularies including Noun, Verb, and relevant explanation. Therefore, Natural language understanding (NLU) system is involved. However, we will not discuss specific methodology or tools about NLU since it is another research issue. The only rule here is to reduce redundancy after extraction. The result structure of vocabularies is organised as below,

Noun: Explanation	Verb: Explanation
...	...

**Table 1: Format of a Set of Vocabulary**

### 3.2 Ontologies Classification

There are a number of terms to be used to classify ontologies. There are Lightweight ontologies that only consist "if" concepts and their relationships, but without many axioms, additional conditions and restrictions; Application Ontologies (AO) contain the definitions specific to a particular application [9], while Reference Ontologies (RO) focus on clarifying the intended meaning of terms used in specific domains. RO is designed for general e-learning system based on LTSA.

The standard of Learning Technology (LTSA) [10] specifies a high level architecture for information technology-supported learning, education and training System component is an important support in this phase.

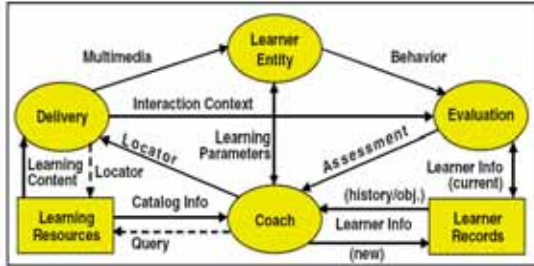


Figure 2: The LTSA System Component [11]

Based on above component structure, an RO is designed and Figure 3 depicts the RO's concepts and relationships.

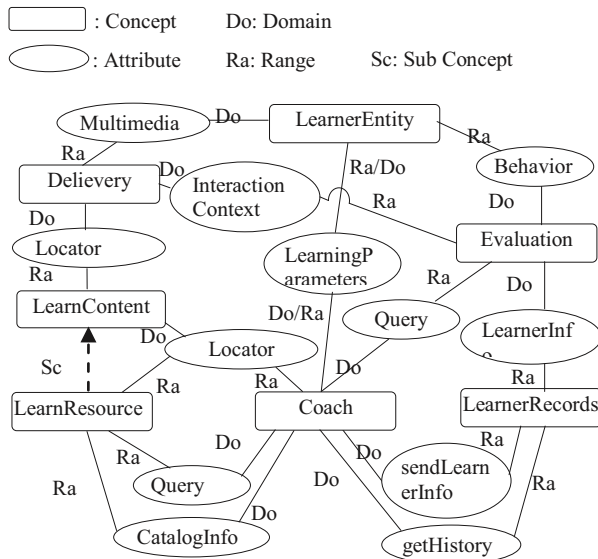


Figure 3: LTSA-based Reference Ontology

The notion of RO is described as a 3-tuple  $RO=(C, A, Sc)$ , where:  $C=Concept$ ,  $A=Attribute$ ,  $Sc=SubConcept$ . Attribute owns a specific definition  $A=(Do, Ra)$ , where:  $Do=Domain$ ,  $Ra=Range$ .

There are three steps to generate AO:

- (1) To generate AO by mapping vocabularies into RO;
- (2) To add extra vocabularies into ontologies as an AO;
- (3) To reduce redundancies for AO.

The notation of AO is described as a 4-tuple  $AO=(C, A, O, Sc)$  that  $A=Attribute$ ,  $O=Object=(Name, Domain, Range, Value)$ ,  $Sc=SubConcept$ . There are more details for e-learning system on Table 2. Besides, the result is a 'good' AO following Gruber's criteria [12] which describes what 'good' ontology should meet: terms clarity, axioms coherence, extensibility, and suitability.

Concepts [C]	Attributes(Range) [A(Ra)]	SubConcept [Sc]
<i>LearnerEntity</i>	leID (String); lePassword(String);	

	Login(Boolean); Logout(Boolean); Multimedia(Delivery); LearningParameters(Coach);	
<i>LearnerRecords</i>	lroID(String); learnerInfo(Evaluation).	
<i>LearnResource</i>	lrID(String); lrContents(X <sup>n</sup> );	LearnContent
<i>LearnContent</i>	lcID(String); lcBegin(X); lcEnd(X).	
<i>Delivery(GUI)</i>	deID(String); locator(LearnContent);	
<i>Evaluation</i>	LearnerInfo(LearnerRecord); Evaluate(LearnerEntity);	
<i>Coach</i>	coID(String); coPassword(String); Login(Boolean); Logout(Boolean); sendLearnerInfo(LearnerRecord); getHistory(LearnerRecord); LearningParameters(LearnerEntity); Locator(Coach, LearnResource); Query(LearnResource); CatalogInfo(LearnResource);	

Table 2: RO List for General E-Learning System

The followings are the specific classification steps:

- (1) To map vocabularies into AO.

In the vocabularies, noun maps to Concept, subConcept, or Object under its explanation. Verb maps to Attribute only. This step results in Table 3 as below:

Concepts [C]	Attributes(Range) [A(Ra)]	Object(O)	SubConcept [Sc]
N1	V1; V2; V3	N11	...
...	...	...	...

Table 3: AO of Step (1)

- (2) To add extra vocabularies into ontologies.

If there are some vocabularies left, a step to add them properly in AO is necessary. The Table 4 shows the Result of step2, where Ex is Extra Vocabulary.

Concepts [C]	Attributes(Range) [A(Ra)]	Object(O)	SubConcept [Sc]
N1	V1; V2; V3	N11	...
...	...	...	...
Ex_N1	Ex_V	Ex_N2	Ex_N3
...	...	...	...

Table 4: AO of Step (2)

- (3) To reduce redundancy for AO.

There are many reasons to introduce redundancy, such as synonyms, verb and noun with same meaning,

vocabularies under inclusive relationship, etc. Table 5 shows the result of this step with strikethrough on redundancy.

Concepts [C]	Attributes(Range)[A(Ra)]	Object(O)	SubConcept [Sc]
N1	V1; V2; V3	N11	
...	...	...	...
<del>Ex_N1</del>	Ex_V	Ex_N2	Ex_N3
...	...	...	...

Table 5: AO of Step (3)

### 3.3 Transform into PIMs

To generate PIMs from AO, a set of transformation rules are proposed.

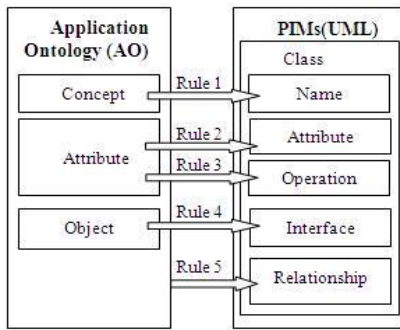


Figure 4: Transformation Rules -- AO to PIMs

Notations of the transformation rules are defined as follows.

- AO=(C, A, O, Sc); C=Concept, A=Attribute=(Name, Domain, Range, Value), O=Object=(Name, Domain, Range, Value), Sc=SubConcept.
- M=PIM=(Cl, In, Re(Clx)). Cl=Class=(Na, At, Op), where, Na=Name, At=Attribute, Op=Operation; In=Interface=(Na, At, Op) where Na=Name, At=Attribute, Op= Operation; Re(Clx)=Relationship with Classes Clx, including 'Generalization', 'Association', and 'Composition'.

#### Rule 1: Mapping Class

In AO, each Concept maps to a Class in PIM. Because every Concept is a noun, Class's name simply is valued by Concept.

```
M.Cl{
  Check AO.C;
  Force M.Cl.Na=AO.C;
}
```

#### Rule 2: Mapping mandatory attributes

In AO, if Attribute's Range is not any Concept or Object, it equals to Class's mandatory attribute.

```
M.Cl.At{
  Check AO.A;
  If(Not AO.A.Ra==AO.C and AO.O){
    M.Cl.At=AO.C.O.A;
  }
}
```

#### Rule 3: Mapping Operations

In AO, if Attribute's Value is verb, it maps to Option of Class.

```
M.Cl.Op{
  Check AO.A;
  If(AO.A.Value==Verb){
    M.Cl.Op=AO.A;
  }
}
```

#### Rule 4: Mapping Interfaces

In AO, if Attribute's Range is a Concept, AO's Object is valued as an Interface in Class.

```
M.In{
  Check AO;
  If AO.A.Range==AO.C{
    M.In==AO.O;
  }
}
```

#### Rule 5: Mapping Relationships

In PIM, three relationships are necessary: Generalization, Association, and Composition. They are mapped separately.

##### (1) Generalisation

Generalisation is 'a-kind-of' relationship. Checking AO, if SubConcept is not empty, there must have a generalisation relationship. One Class is valued by SubConcept which is generated from the other Class that valued by Concept.

```
M.Re{
  Check AO.Sc;
  If(Not AO.Sc==None){
    M.Re=Generalization(M.Cl1==AO.C, M.Cl2==AO.Sc);//Cl2
    generated from Cl1.
  }
}
```

##### (2) Association

Association is a kind of semantic relationship between classes. In AO, if value of one Attribute is another Concept, the mapped classes are associated.

```
M.Re{
  Check AO.A;
  If(have(AO1.A.Value==AO2.C)){
    M.Re=Association(M.Cl1==AO1.C, M.Cl2==AO2.C);//Cl1
    associate with Cl2.
  }
}
```

##### (3) Composition

Composition is a particular association relationship showing components. If the value of an AO's Attribute is a sum of many other AO's Concept, this AO valued Class is composed by other Classes that mapped from those other AO's Concepts.

```
M.Re{
  Check AO.A;
  If(have(AO1.A.Value==AO2.C+AO3.C+...+Aon.C)){
    M.Re=Composition(M.Cl1==AO1.C, M.Cl2==AO2.C...,
    M.Cln=Aon.C);
    //Cl1 composed by Cl2, Cl3, ..., and Cln.
  }
}
```

Following above rules, AO can be transformed into PIM as UML diagrams. Most Classes are mapped from AO's Concepts under Rule 1. Mandatory attributes and operations come from AO's Attribution by Rule 2 and 3. AO's Object been led to Interface following rule 4. Also, there are main relationships generated from AO

depends on rule 5. Particularly, in rule 5, Generalization, Association, and Composition are the three necessary relationships we considered. To sum up, PIM is transformed from AO under proposed rules.

#### 4. Implementation

The Pedagogical ElectroAcoustic Resource Sit (EARS II), a music learning system, is demonstration project in this research, showing the implementation of the proposed approach.

##### (1) Vocabulary Extraction

A piece of the customers' original requirement is showed as follows, which comes from the EARS II music-learning system by Professor Landy [13].

*The pedagogical strategy that is being modelled is a holistic one. It works as follows: there is a three-way approach that is to be presented interdependently. It consists of a 'section' concerning music appreciation ('listening'), one focusing on the understanding of musical, theoretical and technological concepts ('understanding') and another involved with music making ('doing'). The heart is the understanding section as any learner-driven navigation starts here as all key terms and concepts are embedded in this section. [13]*

Extraction is done manually- picked key nouns and verbs- and the result is shown as below,

Noun	Verb
Music appreciation.	Listen
Understanding	Make music
Concept	
Term	
Learner	
Navigation	
Doing	

**Table 6: Extracted Requirement Vocabularies**

##### (2) Ontologies Classification

Step1: To generate AO by mapping vocabularies into RO.

Concepts [C]	Attributes(Range) [A(Ra)]	SubConcept[Sc]
<i>Learner</i> == <i>LearnerEntity</i>	leID (String); lePassword(String); lrID(String).	
<i>Music</i> == <i>LearnResource</i>	lrID(String); lrContents(X);	LearnContent
<i>term, concept</i> == <i>LearnContent</i>	lcID(String); lcBegin(X); lcEnd(X).	
<i>Delivery(GUI)</i>	deID(String); locator(LearnContent);	Listening/Understanding/Doing

**Table 7: AO of Step (1)**

Step 2: To add extra vocabularies into ontologies to be an AO.

In Table 7, Navigation is an extra vocabulary from requirement. Besides, based on LTSA, there is a potential vocabulary, LearnerRecord.

Noun: Explanation
<b>Learner Record:</b> to record Learner's information.
<b>Navigation:</b> to navigate the learning route.

**Table 8: Extra Vocabularies**

Table 9 is the AO with above vocabularies.

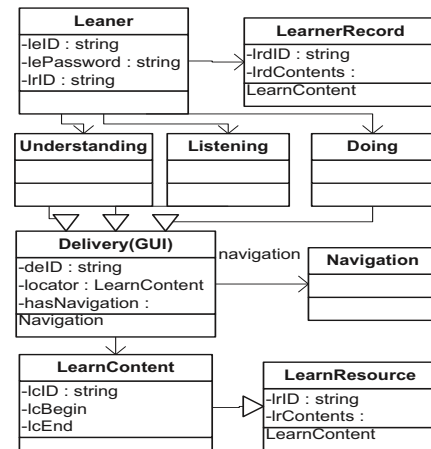
Concepts [C]	Attributes(Range) [A(Ra)]	SubConcept[Sc]
<i>Learner</i>	leID (String); lePassword(String); lrID(String).	
<i>LearnerRecords</i>	LrdID(String); lrdContents(X).	
<i>LearnResource</i>	lrID(String); lrContents(X);	LearnContent
<i>LearnContent</i>	lcID(String); lcBegin(X); lcEnd(X).	
<i>Delivery(GUI)</i>	deID(String); locator(LearnContent); hasNavigation( Navigation ).	Listening/Understanding/Doing
<i>Navigation</i>		

**Table 9: AO of Step (2)**

Step 3: Reduce redundancies for AO. After check, there is no redundancy in Table 9, so it is the final AO for EARS II.

##### (3) Transforming into PIMs

Following transformation rule in section 3.3, a PIM is generated from AO of EARS II as a UML diagram.



**Figure 5: Generated PIM for EARS II**



Figure 5 shows PIM is generated properly. Under the proposed rules, classes, attributes, operations, and relationships are transformed from AO successfully.

## 5. Conclusion and Further Investigation

This paper proposes an ontology-based model driven approach for a music learning system - EARS II. It provides a method to formulate effective platform independent architecture with pedagogical knowledge. Also, this approach supplies the gap between software developers and education experts. The important contributions of this research are that:

- (1) An integrated approach is proposed to guide PIM modelling.
- (2) In designed RO, LTSA and ontology are combined to provide pedagogical support.
- (3) Transformation from ontology to PIM is supported by five rules. Hence, there is a procedure to follow in modelling process.
- (4) An implementation is performed regarding the EARS II project. It evaluates the proposed approach and proves its feasibility and applicability.

The proposed approach is promising to be applied in general e-learning system. Therefore, further work is concerned with automatic transformation, PSMs generation, and improved implementation in EARS II.

## References

- [1] Music, Technology and Innovation Research Centre at De Montfort University, July 2007. "Index: Electroacoustic Music (Genres & Categories [G&C])," [Online]. Available: <http://www.ears.dmu.ac.uk/spip.php?rubrique125>. [Access: January 17, 2009].
- [2] L. Landy, "The ElectroAcoustic Resource Site (EARS) Approaches Its Next Phase: Going global and addressing the young," in *International Computer Music Conference*, 2007, pp. 141-144.
- [3] A. Gavras, M. Belaunde and L. F. Pires and J. P. A. Almeida, "Towards an MDA-Based Development Methodology," in *Lecture Notes in Computer Science: Software Architecture*. Vol.3027. Berlin: Springer, 2004. pp. 230-240.
- [4] D. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley & Sons Inc, 2003.
- [5] F. Chen, "Model Driven Software Modernisation." vol. PhD Leicester: De Montfort University, 2007.
- [6] F. Chen, B. Qiao, H. Yang and W. C. Chu, "A Formal Model Driven Approach to Dependable Software Evolution," in *30th IEEE International Computer Software and Application Conference (COMPSAC'06)*. Vol.1. Chicago, 2006. pp.205-212.
- [7] F. Solms and D. Loubser, "Generating MDA's Platform Independent Model using URDAD," *Knowledge-Based Systems*, vol. 22, 2009. pp. 174-186.
- [8] Y. Tian, H. Yang and L. Landy, "MDA-based Development of Music Learning System," in *14th International Conference on Automation & Computing*, Uxbridge, England, 2008. pp.97-102.
- [9] N. Guarino, "Understanding, Building and Using Ontologies," *International Journal of Human Computer Studies*. vol. 46. Academic Press, 1997. pp. 293-310.
- [10] IEEE Learning Technology Standards Committee, IEEE Computer Society, *IEEE Standard for Learning Technology -- Learning Technology Systems Architecture (LTSA)*. vol. IEEE Std 1484.1, 2003.
- [11] F. Farance and J. Tonkel, "*Learning Technology Systems Architecture (LTSA)*." vol. 1484.
- [12] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human Computer Studies*. vol. 43: Academic Press, 1995. pp. 907-928.
- [13] L. Landy, "The ElectroAcoustic Resource Site (EARS)," *Journal of Music Technology and Education*, 2007. pp. 69-81.
- [14] J. Miller and J. Mukerji, *MDA Guide Version 1.0. 1. Object Management Group: Needham*, 2003.
- [15] J. D. Poole, "Model-Driven Architecture: Vision, Standards and Emerging Technologies", *Workshop on Metamodeling and Adaptive Object Models*, ECOOP, 2001.
- [16] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security", *Lecture notes in computer science*. Springer, 2002, pp. 426-441.
- [17] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*: Addison-Wesley, 2003.
- [18] F. Chen and H. Yang, "Model Oriented Evolutionary Redocumentation," in *IEEE Computer Software and Application Conference (COMPSAC'07)*, Beijing, China, 2007.
- [19] D. Oberle, "Semantic Management of Middleware," ACM New York, NY, USA, 2004. pp. 299-303.
- [20] H. Yang, *Advances in UML and XML-based Software Evolution*. London: Idea Group Pub., 2005.
- [21] D. Gašević, *Model Driven Architecture and Ontology Development*. Berlin: Springer, 2006.
- [22] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowledge Engineering Review*, 1996.
- [23] L. Landy, "Electroacoustic Music Studies and Accepted Terminology: You Can't Have One Without The Others," in *EMS06*, Beijing, 2006.
- [24] L. Yu, J. Zhou, Y. Yi, P. Li, and Q. Wang, "Ontology Model-Based Static Analysis on Java Programs" in *International Computer Software and Applications Conference*, 2008. pp. 92-99.
- [25] F. Chen, S. Li and H. Yang, "A Model Driven Approach to Evolutionary Redocumentation," in *31st IEEE International Computer Software and Application Conference (COMPSAP'07)*, Beijing, China, 2007.

<sup>i</sup> 'X' indicates range is uncertain but depends on specific system.

# Reviewers' Index

## A

Alain Abran  
Silvia Teresita Acuna  
Taiseera Albalushi  
Edward B. Allen

## B

Doo-Hwan Bae  
Ebrahim Bagheri  
Rami Bahsoon  
Xiaoying Bai  
Maria Teresa Baldassarre  
Purushotham Bangalore  
Muhammad Ali Barbar  
Emese Bari  
Daniel Beimborn  
Nicolas Belloir  
Alessandro Bianchi  
Jim Bieman

## C

Danilo Caivano  
Gerardo Canfora  
Joao W. Cangussu  
Giovanni Cantone  
Jeffrey C. Carver  
Garcia-Castro  
Jaelson Castro  
Christine W. Chan  
Keith C.C. Chan  
W.K. Chan  
Kuang-Nan Chang  
Ned Chapin  
Shu-Ching Chen  
Yinong Chen  
Yung-Pin Cheng  
Yoonsik Cheon  
Peter J. Clarke  
Nelly Condori F.  
Panos Constantopoulos

Daniel Cooke

Kendra Cooper  
Maria Francesca Costabile  
Karl Cox  
Juan J. Cuadrado-Gallego  
Alfredo Cuzzocrea

## D

Deepak Dhungana  
Jin Song Dong  
Jing Dong  
Dirk Draheim  
Philippe Dugerdil  
Reiner Dumke  
Schahram Dustdar

## E

Christof Ebert  
Raimund K. Ege  
Faezeh Ensan  
Onyeka Ezenwoye

## F

Davide Falessi  
Behrouz Homayoun Far  
Robert Feldt  
Eduardo B. Fernandez  
Renata Fortes

## G

Kehan Gao  
Alessandro Garcia  
Felix Garcia  
Holger Giese  
Itana Gimenes  
Swapna Gokhale  
Wolfgang Golubski  
Des Greer

Eric Gregoire

Heiko Ludwig  
Michael R. Lyu

## H

Mark Harman  
Xudong He  
Rattikorn Hewett  
Mong-Fong Horng  
Mei Hsing  
Shihong Huang  
Byung-Yeon Hwang

## I

Ali Idri  
Peter In

## J

Clinton Jeffery  
Nils Joachim  
Natalia Juristo

## K

Audris Kalnins  
Sascha Konrad  
Gunes Koru  
Nicholas A. Kraft  
Vinay Kulkarni  
Gi-Hwon Kwon

## L

Mark Last  
Konstantin Laufer  
Jeff Lei  
Tao Li  
Shih-Hsi Liu  
Xiaodong Liu  
Yan (Jenny) Liu  
Yi Liu  
Hakim Lounis  
Zhongyu (Joan) Lu

## M.

Jose Carlos Maldonado  
Jochen Malinowski  
Antonio Mana  
Emilia Mendes  
Harald Meyer  
Rym Mili  
James Miller  
Ana M. Moreno  
Henry Muccini

## N

Martin Neil  
Allen Nikora  
Elisabetta Di Nitto

## O

Mehmet Orgun

## P

Eric Pardede  
Witold Pedrycz  
Jun Peng  
Massimiliano Di Penta  
Antonio Piccinno  
Alfonso Pierantonio

## R

Damith C. Rajapakse  
Rajeev Raje  
Jose Angel Ramos-Gargantilla  
Sanjay Ranka  
Marek Reformat  
Robert Reynolds  
Daniel Rodriguez  
Ignacio Garcia Rodriguez  
Guenther Ruhe

## S

Samira Sadaoui  
Masoud Sadjadi  
Eng. Sattar B. Sadkhan  
Ramon Sagarna  
Ahmed Salem  
S. Alessandro Sarcia  
Kamran Sartipi  
Peter Sawyer  
Douglas Schmidt  
Andreas Schonberger  
Naeem Seliya  
Tony Shan  
Rajan Shankaran  
Yidong Shen  
Michael Shin  
George Spanoudakis  
Arndt von Staa  
Rajesh Subramanyan

## T

Jeff Tian  
Scott Tilley  
Mark Trakhtenbrot  
Laurence Tratt  
Peter Troger  
Jeffrey Tsai  
Tse-Ming Tsai  
T.H. Tse

## V

Antonio Vallecillo  
Michael VanHilst  
Sira Vegas  
Silvia Regina Vergilio

## W

Huanjing Wang  
Christiane Gresse von Wangenheim  
Tim Weitzel  
Victor Winter  
Eric Wong

Franz Wotawa

## X

Haiping Xu

## Y

Chi-Lu Yang  
Hongji Yang  
Ren-Dar Yang  
Huiqun Yu

## Z

Jing Zhang  
Min-Ling Zhang  
Zhinan Zhou  
Hong Zhu  
Xingquan Zhu  
Eugenio Zimeo  
Andrea Zisman

## Authors' Index

### A

Moussa H. Abdallah, 703  
M.K. Abdi, 122  
Alain Abran, 625  
Mohammad Abu-Matar, 291  
Saeed Abu-Nimeh, 542  
Mohsen Afsharchi, 164  
Hamed Ahmadi, 147  
Syed Nadeem Ahsan, 129  
Adriano Bessa Albuquerque, 661  
Randa A. Al-Dallah, 703  
Fernanda Alencar, 43  
Rola I. Al-Khalid, 703  
Ziad Al-Sharif, 194, 392  
Luis Otavio Alvares, 698  
Nelio Alves, 672  
Kristofor Amundson, 688  
Cesar Andres, 426  
Ibrahim Armac, 603  
Michele Atkinson, 157  
Stefan Augustin, 322  
Lerina Aversano, 509  
Stefan Axelsson, 412  
Ruhaya Ab Aziz, 26

### B

Dejan Baca, 412  
Ana Paula Terra Bacelo, 67  
Anders Back, 460  
Eric Baily, 487  
Glivia Angelica Rodrigues Barbosa, 33  
Marcelo Werneck Barbosa, 33  
Emese Bari, 103  
Mike Barker, 631  
David Barrera, 536  
Ricardo Melo Bastos, 67  
Michael A. Bauer, 592  
Georg Beier, 733  
Ayse Bener, 526, 637  
Helmut Berger, 218  
S. Berhe, 575

Swapan Bhattacharya, 694, 715  
Mirco Bianco, 682  
Stefan Biffli, 222, 233  
Michael Bigrigg, 370  
Marcelo Blois, 206  
Vania Bogorny, 698  
Rob Brennan, 228  
Carson Brown, 536  
Torben Brumm, 552  
Ingrid Buckley, 4

### C

Ernest Cachia, 499  
Bora Caglayan, 637  
Ju Cai, 350  
Javier Cano, 676  
Alexandre Cardoso, 672  
Kathleen M. Carley, 370  
Jaelson Castro, 43  
Michele Ceccarelli, 252  
Elizabeth Cecena, 676  
Orieta Celiku, 370  
Peggy Cellier, 432  
Matilde Celma, 676  
Nabendu Chaki, 694  
Hung-Fu Chang, 493  
Lily Chang, 189  
Yeim-Kuan Chang, 338  
Ching-Ming Chao, 530  
E-Liang Chen, 334  
Feng Chen, 569, 739  
Shu-Ling Chen, 306  
Tsong Yueh Chen, 418  
Wei Chen, 406  
Yinong Chen, 280  
Yixin Chen, 442  
Zhenyu Chen, 422  
Sheng-Tzong Cheng, 364  
Yuh-Ming Cheng, 334  
Sucharita Chinchankar, 93  
Chih-Lun Chou, 364

Sankhayan Choudhury, 694  
Tszyan Chow, 280  
Laurel Christian, 487  
Chih-Ping Chu, 338  
Diana Ciric, 548  
Pedro J. Clemente, 721  
Nelly Condory, 676  
Jose M. Conejero, 721  
Jonathan Cook, 438  
Juan Jose Cuadrado Gallego, 625

### D

Andrew D. da Costa, 212  
Cristiano C. da Rocha, 592  
Tiago Santos da Silva, 110  
Viviane T. da Silva, 212  
Caio Stein D'Agostini, 587  
Kamini Dandapani, 103  
Maya Daneva, 73  
M. A. R. Dantas, 592  
Gargi Dasgupta, 16  
Eduardo Santana de Almeida, 328  
Flavia Braga de Azambuja, 67  
Deise de Brum Saccol, 564  
Paloma de Juan, 246  
Jose Luis de la Vara, 55  
Silvio Romero de Lemos Meira, 328  
Carlos J. P. de Lucena, 212  
Renata de Matos Galante, 564  
Marian Fernandez de Sevilla, 619  
Jano Moreira de Souza, 110  
Alessandro De Stasio, 252  
Matthew Del Buono, 487  
S. Demurjian, 575  
Dwight Deugo, 536  
Jose Jorge Lima Dias Junior, 328  
Maria J. Dominguez-Alda, 619  
Antonio Donatiello, 252  
Jin Song Dong, 406  
Derek Doran, 97  
Lucas Drumond, 558  
Weichang Du, 312  
Mireille Ducasse, 432

Sheryl Duggins, 157

### E

Hanan Elazhary, 382  
Hugo Estrada, 61  
Onyeka Ezenwoye, 16

### F

Sandra Fabbri, 386  
Xiaocong Fan, 200  
Behrouz H. Far, 164  
Robert Feldt, 412, 460  
Zaiwen Feng, 178, 286  
Eduardo B. Fernandez, 4  
Sebastien Ferre, 432  
Derek Ferris, 487  
Javed Ferzund, 129  
Renato Fileto, 587  
Lance Fiondella, 480  
Liana Fong, 16  
Josef Froschauer, 218

### G

Renata Galante, 727  
Jerry Gao, 103, 466  
Faiez Gargouri, 258, 613  
David Garlan, 370  
Vahid Garousi, 141  
Markus Gartner, 218  
Enrica Gentile, 581  
Rosario Girardi, 558  
Swapna S. Gokhale, 97, 480  
Wolfgang Golubski, 733  
Xufang Gong, 456  
Lilia Grati, 613  
He Guo, 569  
Jin-gang Guo, 505

### H

Jafar Habibi, 147

Fady Hamoui, 597  
Tobias Haubold, 733  
Michael Hausenblas, 240  
Akihiro Hayashi, 666  
Keqing He, 178  
Xiao-yang He, 505  
Xudong He, 189  
Matthew J. Henderson, 520  
Peter Henderson, 520  
Juan Hernandez, 721  
Miguel A. Herranz, 619  
Carlos A. Heuser, 698  
Mong-Fong Horng, 334  
Sam Hsu, 93  
Gang Huang, 446  
Jianchu Huang, 262  
Shihong Huang, 93, 514  
Wen-Shyang Huang, 334  
Zhiqiu Huang, 274  
Marianne Huchard, 597

### I

Carlos A. Iglesias, 246  
Magda G. Ilieva, 49  
Katsuro Inoue, 631  
Roberto Intonti, 509  
Aftab Iqbal, 240

### J

Clinton Jeffery, 194, 392  
Changbin Ji, 422  
Shunhui Ji, 350, 456  
Yuting Jiang, 350  
Wenpin Jiao, 152  
Elena Jurado, 721

### K

Denis Kacan, 412  
Selim Kalayci, 16  
Ali Kamandi, 147  
Jian Kang, 262

Omar Abou Khaled, 709  
Taghi M. Khoshgoftaar, 81, 116, 344  
Ekrem Kocaguneli, 526, 637  
Lingjun Kong, 37  
Sven J. Korner, 552  
Nicholas A. Kraft, 268  
Dhananjay Kulkarni, 548  
Izuru Kume, 376  
Fei-Ching Kuo, 418

### L

Edgard Lamounier, 672  
Leigh Landy, 739  
Heesang Lee, 170  
Jen-Kuin Lee, 334  
Ana Paula Lemke, 206  
Michael Leuschel, 400  
Bing Li, 286  
Bixin Li, 350, 456  
Han Li, 569  
Jianzhi Li, 262, 569  
Juan Li, 37  
Xiang Li, 274  
Yin Li, 37  
Yan Liang, 268  
Bin-Yi Liao, 334  
Pengpeng Lin, 81  
Chien-Hung Liu, 306  
Huai Liu, 418  
Ruimin Liu, 569  
Yang Liu, 406  
Yanhong A. Liu, 406  
H. Lounis, 122  
Stephen C-Y. Lu, 493  
Marcia Lucena, 43  
Simone A. Ludwig, 688

### M

Jia-kuan Ma, 505  
Yutao Ma, 178  
Dawn MacIsaac, 312  
Jihen Majdoubi, 258

Dipankar Majumdar, 715  
Heloise Manica, 592  
Borja Martin, 619  
Alicia Martinez, 61  
Ken-ichi Matsumoto, 631  
Tomoko Matsumura, 631  
Tom McBride, 26  
Jim McElroy, 649  
Nancy R. Mead, 542  
Hong Mei, 446  
Manoel Mendonca, 386  
Mercedes G. Merayo, 426  
Dieter Merkl, 218  
Mark Micallef, 499  
Yoshiki Mitani, 631  
Seiya Miyazaki, 542  
Shahrouz Moaven, 147  
Hossein Momeni, 609  
Akito Monden, 631  
Richard Mordinyi, 222  
Mirella M. Moro, 564  
Thomas Moser, 222, 233  
Javed Mostafa, 185  
Elena Mugellini, 709  
Snehasis Mukhopadhyay, 185  
John Mylopoulos, 61

## N

Manfred Nagl, 603  
Shin Nakajima, 20  
Xiaofei Nan, 442  
Valeh H. Nasser, 312  
Balduino F. dos S. Neto, 212  
Manoel T. de A. Netto, 212  
Nicole Novielli, 581  
Manuel Nunez, 426  
Amjad Nusayr, 438

## O

Gabriel Oliveira, 698  
Olga Ormandjieva, 49  
Declan O'Sullivan, 228

## P

Sergio Paim, 672  
Mathew Palakal, 185  
Jeng-Shyang Pan, 334  
Weifeng Pan, 286  
Abhijit Pandya, 93  
Marilyn Parker, 93  
Oscar Pastor, 61  
Pushkala Pattabhiraman, 103  
Rong Peng, 178  
Shengquan Peng, 185  
Tao Peng, 286  
Xin Peng, 135  
Beatriz Perez, 318  
Fredrik Petterson, 460  
Marcelo S. Pimenta, 727  
Macario Polo, 318  
Daniel Porto, 386  
Sarah Printy, 487

## Q

Dong Qiu, 350, 456

## R

Zornitza Racheva, 73  
Vahid Rafe, 609  
Adel T. Rahmani, 609  
Rajeev Raje, 185  
Krishna Ratakonda, 643  
Pedro Reales, 318  
P.Krishna Reddy, 87  
Daniel Retkowitz, 603  
Olivier Ridoux, 432  
Jorge Rivas, 676  
Ana Regina Rocha, 661  
Sergio Assis Rodrigues, 110  
Roberto Rodriguez, 721  
Gustavo Rossi, 4  
Guenther Ruhe, 141, 649  
Gaurav Ruhela, 87  
Vasile Rus, 442



## S

S. Masoud Sadjadi, 4, 16  
H. Sahraoui, 122  
Salamah Salamah, 487  
Mireille Samia, 400  
Juan Sanchez, 55  
Emanuel Santos, 43  
R. Saripalle, 575  
Anirban Sarkar, 694  
Shailashree Savanur, 466  
Gregor Scheithauer, 322  
Bradley Schmerl, 370  
Ingo Seidel, 218  
Naeem Seliya, 116  
Taemin Seo, 170  
Maulik Shah, 103  
Mihir Shah, 103  
Ali Shahrokni, 460  
Mohsen Sharifi, 609  
Chih-Chin Shen, 530  
Liwei Shen, 135  
Etsuya Shibayama, 376  
Jiashing Shih, 364  
Michael E. Shin, 10  
Sajjan Shiva, 442  
Darius Sidlauskas, 412  
Alberto Sillitti, 682  
Carla Silva, 43  
Fabio Silva, 558  
John C. Sloan, 344  
Randy K. Smith, 268  
Maria Sokhn, 709  
Hui Song, 152  
Pablo R. Soria, 619  
Giancarlo Succi, 682  
Jun Sun, 406  
Yanchun Sun, 152, 446  
Kiran Gopala Reddy Sunanda, 10  
Wikan Danar Sunindy, 222, 233

## T

Dan Tappan, 295  
Hendrik Thomas, 228

Craig Thompson, 688  
Yingchun Tian, 262, 739  
Mohamed Tmar, 258, 613  
Jose Leomar Todesco, 592  
Alexandre Torres, 727  
Maria Tortorella, 509  
Ayse Tosun, 526, 637  
W.T. Tsai, 280  
Giovanni Tummarello, 240  
Burak Turhan, 637

## U

Muhammad Irfan Ullah, 141  
Oana Ureche, 240  
Christelle Urtado, 597

## V

Michael VanHilst, 514  
Augusto Varas, 344  
Sylvain Vauttier, 597  
Hema Veeraragavathatham, 466  
Balaji Viswanathan, 16  
Dante Vitale, 252  
Devarshi Vyas, 103

## W

Bo-Sian Wang, 334  
Huanjing Wang, 81  
Jian Wang, 178  
Qing Wang, 37, 655  
Yasha Wang, 356, 505  
Yongji Wang, 655  
Yuxin Wang, 569  
Xiao Wei, 280  
Rose Williams, 643  
Guido Wirtz, 322  
Franz Wotawa, 129  
Hong-Chi Wu, 334  
Mingzoo Wu, 364  
Weibiao Wu, 473

## **X**

Jinchun Xia, 466  
Junchao Xiao, 655  
Bing Xie, 356  
Lizi Xie, 655  
Baowen Xu, 422  
Zhiwei Xu, 473

## **Y**

Chi-Lu Yang, 338  
Hongji Yang, 262, 569, 739  
Lili Yang, 456  
Ye Yang, 37, 655  
Zilan Yang, 164  
Huilin Ye, 301  
Min Yuan, 274

## **Z**

Leopoldo Zepeda, 676  
Du Zhang, 450  
Shao Jie Zhang, 406  
Jian Zhao, 274  
Junfeng Zhao, 356  
Wenyun Zhao, 135  
Zhihong Zhao, 422  
Cheng Zhong, 164  
Hong Zhou, 569  
Wu Zhou, 505  
Zhou Zhou, 473  
Wenhui Zhu, 446  
Didar Zowghi, 26  
Fernanda Zulkarnain, 548



# SEKE 2010 Call For Papers

## The Twenty-Second International Conference on Software Engineering and Knowledge Engineering

Hotel Sofitel, San Francisco Bay, USA  
July 1 - July 3, 2010

Organized by  
**Knowledge Systems Institute Graduate School**

The Twenty-Second International Conference on Software Engineering and Knowledge Engineering (SEKE'10) will be held at the Hotel Sofitel, Redwood City, California, USA, July 1-3, 2010.

The conference aims at bringing together experts in software engineering and knowledge engineering to discuss on relevant results in either software engineering or knowledge engineering or both. Special emphasis will be put on the transference of methods between both domains.

### TOPICS

Agent architectures, ontologies, languages and protocols  
Multi-agent systems  
Agent-based learning and knowledge discovery  
Interface agents  
Agent-based auctions and marketplaces  
Artificial life and societies  
Secure mobile and multi-agent systems  
Mobile agents  
Mobile Commerce Technology and Application Systems  
Mobile Systems

Autonomic computing  
Adaptive Systems  
Integrity, Security, and Fault Tolerance  
Reliability  
Enterprise Software, Middleware, and Tools  
Process and Workflow Management  
E-Commerce Solutions and Applications  
Industry System Experience and Report

Service-centric software engineering  
Service oriented requirements engineering  
Service oriented architectures  
Middleware for service-based systems  
Service discovery and composition  
Quality of services  
Service level agreements (drafting, negotiation, monitoring and management)  
Runtime service management  
Semantic web

Requirements Engineering  
Agent-based software engineering  
Artificial Intelligence Approaches to Software Engineering  
Component-Based Software Engineering  
Automated Software Specification  
Automated Software Design and Synthesis  
Computer-Supported Cooperative Work  
Embedded and Ubiquitous Software Engineering  
Measurement and Empirical Software Engineering  
Reverse Engineering  
Programming Languages and Software Engineering  
Patterns and Frameworks  
Reflection and Metadata Approaches  
Program Understanding

Knowledge Acquisition  
Knowledge-Based and Expert Systems  
Knowledge Representation and Retrieval  
Knowledge Engineering Tools and Techniques  
Time and Knowledge Management Tools  
Knowledge Visualization  
Data visualization  
Uncertainty Knowledge Management  
Ontologies and Methodologies  
Learning Software Organization  
Tutoring, Documentation Systems

Human-Computer Interaction  
Multimedia Applications, Frameworks, and Systems  
Multimedia and Hypermedia Software Engineering

Smart Spaces  
Pervasive Computing  
Swarm intelligence  
Soft Computing

Software Architecture  
Software Assurance  
Software Domain Modeling and Meta-Modeling  
Software dependability  
Software economics  
Software Engineering Decision Support  
Software Engineering Tools and Environments  
Software Maintenance and Evolution  
Software Process Modeling  
Software product lines  
Software Quality  
Software Reuse  
Software Safety  
Software Security  
Software Engineering Case Study and Experience Reports

Web and text mining  
Web-Based Tools, Applications and Environment  
Web-Based Knowledge Management  
Web-Based Tools, Systems, and Environments  
Web and Data Mining

### CONFERENCE SITE (HOTEL INFORMATION)

The SEKE 2010 Conference will be held at the Hotel Sofitel, Redwood City, California, USA. The hotel has made available for these limited dates (6/30 - 7/4/2009) to SEKE 2010 attendees a discount rate of \$89 US dollars for single/double, not including sales tax.

### INFORMATION FOR AUTHORS

Papers must be written in English. An electronic version (Postscript, PDF, or MS Word format) of the full paper should be submitted using the following URL: <http://conf.ksi.edu/seke10/submit/SubmitPaper.php>. Please use Internet Explorer as the browser. Manuscript must include a 200-word abstract and no more than 6 pages of IEEE double column text (include figures and references). Workshop papers should be submitted to the workshops directly.

### INFORMATION FOR REVIEWERS

Papers submitted to SEKE'10 will be reviewed electronically. The users (webmaster, program chair, reviewers...) can login using the following URL: <http://conf.ksi.edu/seke10/review/pass.php>.

If you have any questions or run into problems, please send e-mail to: [seke10@ksi.edu](mailto:seke10@ksi.edu).

SEKE 2010 Conference Secretariat  
Knowledge Systems Institute Graduate School  
3420 Main Street  
Skokie, IL 60076 USA  
Tel: 847-679-3135  
Fax: 847-679-3166  
E-mail: [seke10@ksi.edu](mailto:seke10@ksi.edu)

### IMPORTANT DATES

March 2, 2010      Paper submission due  
April 15, 2010     Notification of acceptance  
May 10, 2010      Camera-Ready Copy

# 2009 SEKE

Boston, Massachusetts  
July 1-3, 2009

Proceedings of the  
Twenty-First International  
Conference on  
Software Engineering &  
Knowledge Engineering



Copyright © 2009  
Printed by  
Knowledge  
Systems  
Institute  
3420 Main Street  
Skokie, Illinois 60076  
(847) 679-3135  
info@ksi.edu  
www.ksi.edu  
Printed in USA, 2009  
ISBN 1-891706-24-1 (paper)