DAH'11:
International Workshop on Dynamic and Adaptive Hypertext: Generic Frameworks, Approaches and Techniques
ACM Hypertext 2011
22nd ACM Conference on Hypertext and Hypermedia
Eindhoven, June 6-9

Proceedings of the

Second International Workshop on Dynamic and Adaptive Hypertext: *Generic Frameworks, Approaches and Techniques* (DAH'11)

co-located with
22nd ACM Conference on Hypertext and Hypermedia
(Hypertext'11)

2011, Eindhoven, the Netherlands

Editors: Mykola Pechenizkiy[1], Evgeny Knutov[1], Michael Yudelson[2], Fabian Abel[3], Geert-Jan Houben[3], Eelco Herder[4]

[1] Information Systems Group, Department of Computer Science,
Eindhoven University of Technology (TU/e), Eindhoven, The Netherlands
{m.pechenizkiy, e.knutov}@tue.nl

[2] Human Computer Interaction Institute,
Carnegie Mellon University, Pittsburgh, USA
yudelson@andrew.cmu.edu

[3] Web Information Systems,
Delft University of Technology, Delft, The Netherlands
{f.abel, g.j.p.m.houben}@tudelft.nl

[4] Forschungszentrum L3S, Hannover, Germany
herder@l3s.de

Preface

Dynamically generated hypertext adapted and personalized to the users' needs and abilities has proven to be a very successful technique over the last decade and a half. It is particularly helpful for reducing the information overload that frequently occurs in the modern information-driven world. Adaptive hypertext is equally effective for accessing many types of items, be it news, products, artifacts or descriptions thereof in electronic shops, libraries or museums, or even learning materials.

Architecture and framework building efforts allow hypertext community to lay the foundations for the creation of generic system reference models that spawn research activities in multiple domains. Examples of such generic models are AHAM for adaptive hypermedia and FOHM for open hypermedia as well as the APeLS and Personal Reader frameworks for service-based adaptive hypermedia.

Rapid expansion of hypertext, web-based systems, and adaptive hypermedia resulted in the emergence of a plethora of new terms, conceptual models, and prototype systems. Classical hypermedia models are no longer capable of capturing phenomena that evolve in the Social and Semantic Web. In particular, open corpus adaptation, ontologies, group adaptation, and data mining tools for adaptation are not supported or supported in a limited fashion.

The DAH'11 workshop was organized in conjunction with the 22th ACM International Conference on Hypertext and Hypermedia and held on June 6, 2011, in Eindhoven, the Netherlands. It builds on the success of DAH'09 at Hypertext'09 and WABBWUAS'10 workshop at UMAP 2010. The workshop provided a focused international forum for researchers to present, discuss and explore the state of the art as well as outline promising future research directions of dynamic and adaptive hypertext. The workshop addressed different aspects of dynamic and adaptive hypertext by focusing on generic frameworks, approaches and techniques and ways of reusing novel models and/or existing system and their components for building adaptive hypermedia systems. The DAH'11 workshop therefore covered the following (non-exhaustive) list of topics:

- Adaptation and personalization
  - open-corpus adaptation
  - group adaptation
  - sharing user models
- Adaptive/Dynamic Hypertext authoring
  - authoring conceptual adaptation models
- Data mining for
  - user modeling
  - domain modeling
  - automatic generation of adaptation rules
- Adaptation frameworks
  - reusing adaptation reasoning and techniques
  - evaluation of frameworks
  - scalability and performance issues

The DAH'11 proceedings include six accepted contributions that were presented at the workshop.

The first paper by Hannon *et al.* "Bridging Recommendation and Adaptation: Generic Adaptation Framework - Twittomender compliance case-study" discusses Recommender System (RS) modeling in terms of Adaptive Hypermedia Systems (AHS). The authors investigate AHS and RS functionality compliance in terms of common features, functionality, building blocks and composition of the system and bring up complementary aspects of adaptation, personalization and recommendation in a context of a generic framework. As a case study of their research the authors scrutinize the 'Twittomender' RS, decompose it in building blocks, outline and highlight its properties along with the advantages and possible enhancements of the system.

Celik *et al.* present their work entitled "Towards a Framework for Adaptive Faceted Search on Twitter" and propose strategies for inferring facets and facet values on Twitter by enriching the semantics of individual Twitter messages. The paper presents different methods, including personalized and context-adaptive methods, for making faceted search on Twitter more effective. The authors also conduct a preliminary analysis that shows that semantic enrichment of tweets is essential for faceted search on Twitter and that there is essential need for adaptive faceted search on Twitter and finally propose an evaluation methodology.

Grishchenko *et al.* in "Referencing within evolving hypertext" introduce a minimalistic but powerful query language of specifiers that allows for great flexibility in referencing within a changing hypertext. This helps to capture the state of the hypertext, point at changes, expose authorship or blend branches of the versioned hypertext structures. Their approach is based on the Casual Tree model.

The paper by Zemirline *et al*. "A set of adaptation patterns for expressing adaptive navigation in Adaptive Hypermedia" presents a set of 22 adaptation patterns which are independent of any application domain and adaptation engines. These patterns have been translated to LAG and GLAM adaptation languages in order to plug them on existing adaptation engines. Currently they are used in the so-called EAP framework to define complex adaptation strategies.

Knutov *et al.* in "Adaptive Hypermedia Systems Analysis Approach by Means of the GAF Framework" consider an analysis approach of AHS composition and design by defining building blocks' interfaces and presenting corresponding dependencies by means of the GAF framework, which helps to identify system design guidelines and and facilitates the creation of adaptive systems from scratch. In their paper authors analyze adaptive system behaviour, architecture and risks involved.

In the last paper "Open Corpus Adaptation++ in GALE: Friend or Foe?" by Smits and De Bra raise a discussion about Open Corpus Adaptation. They describe how their GALE engine implementation achieves Open Corpus adaptation functionality and pose the question whether this is actually a desired feature or potentially a dangerous addition with unintended consequences.

We would like to thank the authors for their interest in the workshop and for submitting their contributions. And we thank the PC members for their help in reviewing the submitted papers.

Mykola Pechenizkiy
Evgeny Knutov
Michael Yudelson
Fabian Abel
Eelco Herder
Geert-Jan Houben

DAH'11 Organizing Committee, June 2011

Organization
DAH 2011 is co-located with the 22nd International Conference on Hypertext and Hypermedia (HT) 2011, Eindhoven, The Netherlands.

Organizing Committee
Mykola Pechenizkiy
Evgeny Knutov
Michael Yudelson
Fabian Abel
Eelco Herder
Geert-Jan Houben

Program Committee
Paul, De Bra (Eindhoven University of Technology, the Netherlands)
Kevin Koidl (Trinity College Dublin, Ireland)
Ben Steichen (Trinity College Dublin, Ireland)
Ian O'Keeffe (Trinity College Dublin, Ireland)
Riccardo Mazza (SUPSI, University of Lugano, Switzerland)
Tsvi Kuflik (Haifa University, Israel)
Daniel Krause (Leibniz Universität Hannover, Germany)

Table of Contents

Bridging Recommendation and Adaptation: Generic Adaptation Framework - Twittomender compliance case-study.

John Hannon[2], Evgeny Knutov[1], Paul De Bra[1],
Mykola Pechenizkiy[1], Kevin McCarthy[2], and Barry Smyth[2]

[1] Department of Computer Science, Eindhoven University of Technology,
P.O. Box 513, 5600 MB, Eindhoven, the Netherlands
e.knutov@tue.nl, debra@win.tue.nl, m.pechenizkiy@tue.nl
[2] Clarity: Centre for Sensor Web Technologies,
School of Computer Science and Informatics,
University College Dublin, Dublin 4, Ireland
john.hannon@ucd.ie, kevin.mccarthy@ucd.ie, barry.smyth@ucd.ie

Abstract. In this paper we consider Recommender System (RS) modeling in terms of Adaptive Hypermedia Systems (AHS) and investigate AHS and RS functionality compliance in terms of common features, functionality, building blocks and composition of the system. We bring up complementary aspects of adaptation, personalization and recommendation in a context of a generic framework which provides properties of information fusion and heterogeneity and could serve as a reference model. We show major recommendation functionality in terms of the reference structure and recommendation process by presenting a conceptual generic 'adaptation-recommendation' sequence chart which overlays and combines properties of adaptation and recommendations taking advantages of both. In fact we show that RS if implemented on the web can be considered as AHS, in this wise a generic framework should be capable of describing virtually any RS. In the case study we scrutinize the *Twittomender*[3] RS. We decompose the system in building blocks, outline and highlight its properties along with the advantages and possible enhancements. We conclude by summarizing framework advantages and AH recommendation compliant features as well as lessons learned from this study.

1 Introduction

In recent years a lot of progress has been made in the field of recommender systems (RS). New efficient models and algorithms have been developed [8]; heterogeneous and hybrid systems are gaining wide use. Amongst those RS personalized search and twitter are getting into place [4, 9].

Since its first emergence on the scene in early 2006, Twitter has grown from an early microblogging engine into a real time web behemoth. In the early days some 300,000 tweets were produced by early adopters of the system per month, contrast this with some 140 million tweets estimated to be produced per day in march 2011[4]. With this

[3] http://twittomender.ucd.ie/
[4] C. Penner, #numbers, mar, 2011, http://blog.twitter.com/2011/03/numbers.html

obvious information increase, creating ways to use this information appropriately and intelligently has become a hot topic for many researchers [7, 3]. In Twitter the main producers of this content are the users themselves who have subscribed to the system. Obviously, not all the information produced by each user is to everyone's preference, so finding the producers, the so-called "diamonds in the rough", is an interesting research challenge. And, with the *Twittomender* system we have chosen to frame this as a recommender systems.

In the very closely related field of Adaptive Hypermedia (AH) many things have changed since one of the first reference models models AHAM as well: new terms, definitions and models have been introduced and, new adjacent areas (such as the aforementioned RS, personalized web search) have been investigated, new prototypes developed. Most of these AH models, including the emerging and discussed in the paper GAF model focus on a layered architecture and discuss the adaptation of the content and navigation to the user properties as well as recommending the links to follow based on the user preferences and knowledge, thus bringing the fields of AH and Recommendation closer together. In fact we show that a web-based RS can be treated as AHS and therefore expressed in terms of a generic AH framework, which we backup by the conducted case-study of the *Twittomender* system.

In the paper we focus on describing RS functionality and *Twittomender* [4] in particular in terms of AH systems, in order to show complimentary features of RS and layers of a generic AH framework (GAF) which aims to develop a new reference model for the adaptive hypermedia research field by considering new developments, techniques and methodologies in the area of AH and adjacent fields (including but not limited to RS) [5]. We show framework compliance (conformity in structure and partial functionality overlay) with the *Twittomender*, a Twitter based recommender system. Besides *Twittomender* is a very interesting example, covering topics such as personalization, recommendation, search and collaborative web experience which means that we can bring and fit all these features together in the framework. As a result we also achieve an evaluation of how generic the GAF framework could be on the one hand and whether the description of one particular RS fits in the framework, and helps to evaluate the real system on the other.

2 Overview of RS compliance with a generic AHS: system building blocks

The main goal of the *GAF* framework is to provide a reference architecture of AHS, to describe essential and optional elements of an adaptive system, define the criteria to distinguish between these elements, describe their functionality and interaction. Here we define a modular structure (framework) that can be used to describe and develop applications that satisfy different adaptation and recommendation needs. This framework has a layered structure where layers match the original classification of AH methods and techniques and provide the per layer functionality separation 1. Using this classification we describe recommendation functionality within the same adaptation system layers depending on the requirements of the application and thus contribute to the system extensibility and heterogeneity. We show the correspondence of AHS modeling (in particular GAF layers) and recommender approaches. We will first summarize the applicability of GAF with respect to different types of RS and then walk through the framework layers and functionality to show the compliance.

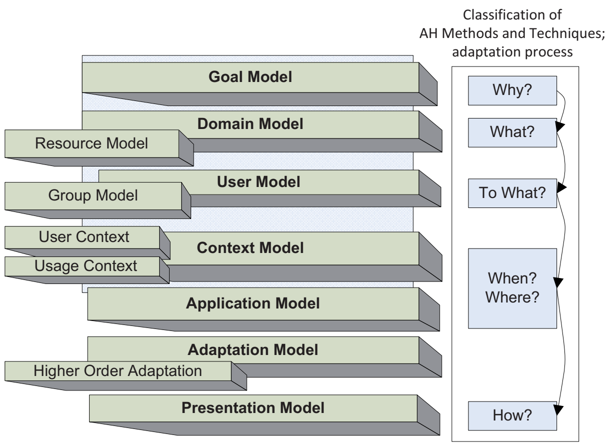


Fig. 1. GAF building blocks

Considering RS we would like to mention the major types of RS and give a brief insight on the GAF structure and functionality related to recommendations:

– *Collaborative Filtering (CF)* is usually based on a User Model (UM) and a Group model (GM) as known also in AHS (we assume here both user-to-user and item-to-item aspects of CF). The recommendations are generated based on a comparison of user profiles from the GM and UM. In this case a Recommender UM is represented by a vector of items and corresponding ratings, which essentially can be turned into a set of *interesting* items or concepts (for this user) and associated attributes with corresponding rating values [6]. Those values are updated as the user interacts with the systems, browses through items, rates them, and receives recommendations. In general Collaborative Filtering allows to deal with different types of objects where essentially only ratings matter which is made possible by separating ratings and item set within the GAF Domain, User and Resource models.
– *Content-based* RS recommend an item to a user based upon a description of the item - feature database (e. g. using words in a text as the textual feature or book genres as library properties) and a user profile (UM). Having learned features of these items (which can be expressed in the Resource Model (RM)) rated by the user, the RS infers new recommendation suggestions. As well as collaborative filtering content-based recommenders can also handle different types of objects as long as there is a common feature space.
– In *Knowledge-based Recommendation* systems a domain expert knows which types of recommended items should be assigned to which types of users. In fact this is in

line with current state of the art in AHS (i. e. there is a domain expert who needs to author UM, DM and Adaptation Rules mapping these two). It combines content-based filtering performed on the features of the concerned dataset with the explicit user query which is used to make inferences about needs and preferences of the the user. Thus it is possible to relate how a particular item meets user needs and thus to do the reasoning about further possible recommendations.
– *Hybrid type Recommender systems* are the most commonly used type. They employ different approaches (simultaneously) to achieve better results, both combining techniques from collaborative, content-based and knowledge-based methods and providing different type of hybridization: mixed, weighted, cascade, etc. Our AHS framework allows us to combine both advantages of content and collaborative filtering, having all necessary building blocks available (e. g. user and group models, rankings mechanisms, reasoning engine) and will also help to handle heterogeneous data sources.

3 Twittomender recommender overview

The *Twittomender* system's main function involves syncing a users account and producing followee recommendations through a range of collaborative and content-based strategies. However for this to work efficiently, users must be active on Twitter, i.e. they must follow a number of other users, must have some followers themselves and must have produced some content (through tweets). Although this functionality is great for Twitter users who wish to increase the number of appropriate user streams they follow, it does not perform satisfactorily for new Twitter users. These users have not produced much content through tweets, nor are they following or being followed by enough users for collaborative or content-based followee recommendation techniques to perform as expected. For this reason we also provide a search capability to *Twittomender*, which allows users to explicitly type search queries. For our collaborative and content-based strategies we evaluate 9 different profiling and recommendation strategies based on the different sources of profile information, in isolation and in combination. To begin with we implemented 4 *content-based* strategies that rely on the content of tweets as follows:

1. (*S1*) users are represented by their own tweets
   ($tweets(U_T)$);
2. (*S2*) users are represented by the tweets of their followees ($followeestweets(U_T)$);
3. (*S3*) users are represented by the tweets of their followers ($followerstweets(U_T)$);
4. (*S4*) a hybrid strategy in which users are represented by the combination of tweets from $tweets(U_T)$,
   $followeestweets(U_T)$, and $followerstweet(U_T)$;

In addition we implemented 3 *collaborative* style strategies, in the sense that we view a user profile as a simple set of user ids.

5. (*S5*) users are represented by the IDs of their followees ($followee(U_T)$);
6. (*S6*) users are represented by the IDs of their followers ($follower(U_T)$);
7. (*S7*) a hybrid strategy in which users are represented by the combination $followee(U_T)$ and $follower(U_T)$;

Additional 2 strategies are:

8. (*S8*) the scoring function is based on a combination of content and collaborative strategies S1 and S6;
9. (*S9*) the scoring function for this strategy is based on the position of the user in each of the recommendation lists so that users that are frequently present in high positions are preferred;

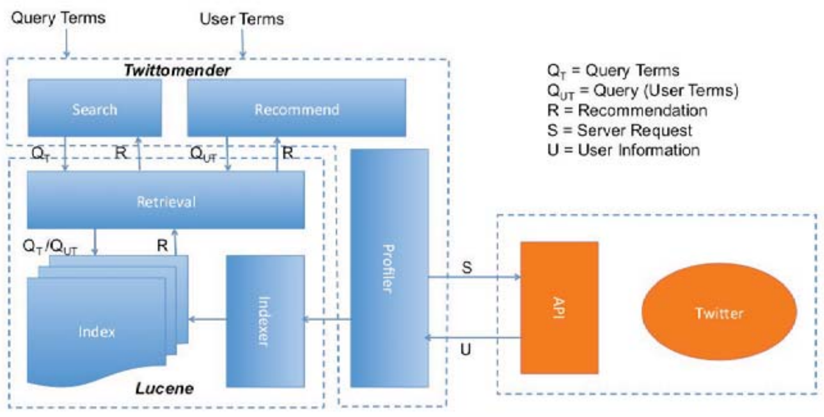


Fig. 2. Twittomender architecture

4 Recommender System and AHS: Twittomender Study

Considering diverse aspects of the system and its functionality we decompose it in a a way that forms an overlay of a generic model of an adaptive system, explains the functionality of the system using terms and definitions from adjacent recommender systems research area, and foremost brings a custom system to a common denominator by means of the GAF reference model.

Fig. 3 presents a picture of *Twittomender* and the Generic Adaptation Process (GAP) *sequence chart* compliance. Here the GAP process chart is constructed by coupling the layers of a general purpose AHS as described in [5]. Recommendation steps are assigned to a single layer or a transition in the system. Though we have faced certain issues distinguishing parts of the Recommendation Engine functionality, in particular the *filtering* and *ranking* mechanisms (in this respect Application Model (AM) and Adaptation Model/Engine (AE) can be treated accordingly) we could align *Twittomender* functionality with GAF terms and identify gaps and possible extensions. On the one hand the mapping proves the genericity of the GAF framework, and on the other hand it opens new horizons to facilitate and generalize recommendation aspects, bringing adaptive techniques into place, extend information fusion and heterogeneity possibilities [2] of such a systems encapsulating features of both Recommender and Adaptive Systems.

Further we summarize compliant and complementary *Twittomender* features and the reference structure of GAF and explain the building blocks and interactions presented in Fig. 3. Of particular interest here are the remarks regarding AHS functionality (shown in GAF terms) that can be used to extend *Twittomender*, but also a few instances where *Twittomender* functionality suggests further extensions to the GAF model.

– Users start system interaction by choosing whether to get recommendations directly by logging in with their Twitter profile or by entering a search query they are interested in. This refers to a Goal Model of GAF. Internally goal is represented by the immediate query input by the user or constructed from the indexed content of the users tweets when he or she log-ins into the system.
– Twittomender Profiler serves both as a UM (User Model), by associating each user with the corresponding group of followers and followees, and at the same time as user information mediator which requests tweet content information from Twitter services and provides this information to Lucene indexer which forms the index of user tweets and such forms the domain model to be used in recommendations.
– The Group Model refers to maintaining a collaborative user profile is already provided by Twitter services. It clusters results by location or user age group and gender, and uses it to rank and recommend results for a particular user or mediate user models associated with different groups. To some extent Twitter services provide this possibility by maintaining the groups of followers, followees of any given user.
– Domain Model (DM) of the *Twittomender* is represented by the index which is stored by the Lucene (backend).
– Context models (both user and usage models) are not considered.
– Application Model is represented by the *Twittomender* framework. Mainly it serves to query terms from the Lucene and retrieve corresponding ranked lists of users and related tweets. *Twittomender* framework also provides interfaces to the Presentation Model (PM).
– Adaptation Model as described in a generic Recommender system use-case is represented by indexing and actual querying solution, Lucene. Its Information Retrieval module provides querying interfaces to *Twittomender* and return recommendation lists upon querying (both User Terms and search Terms as indicated in the Goal Model. The actual index is stored in DM providing flexibility of the system and at the same time decoupling Lucene as a stand-alone Query/Retrieval mechanism.
– Presentation Model generates ranked list of users recommended to follow and corresponding cloud of indexed terms that are relevant to the user activity in Twitter.

5 Advantages of GAF in Recommendations

Based on our *Twittomender* case-study we were able to define the following GAF advantages which can be used to improve and extend the system functionality:

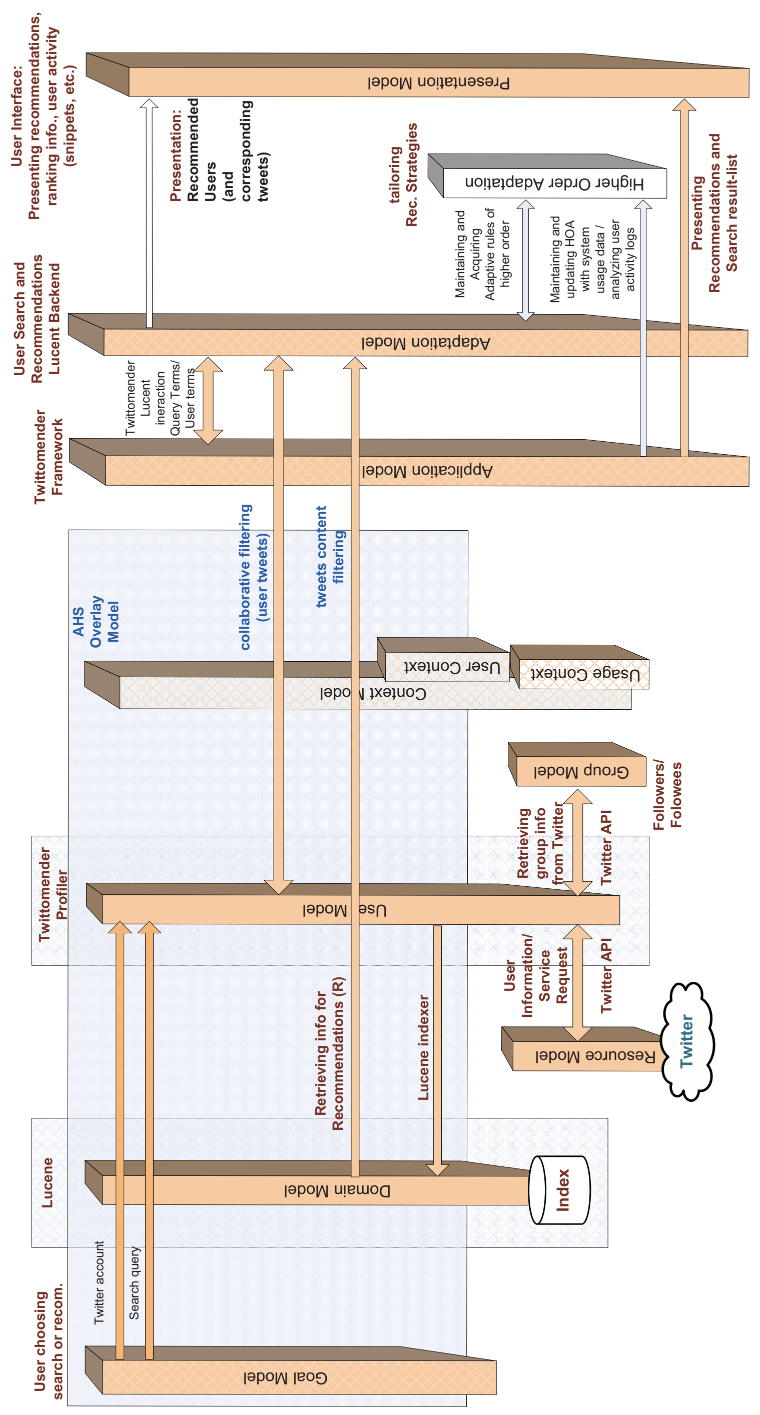

User Interface: Presenting recommendations, ranking info., user activity (snippets, etc.)
Presentation Model
Presentation: Recommended Users (and corresponding tweets)
tailoring Rec. Strategies
Higher Order Adaptation
Maintaining and Acquiring Adaptive rules of higher order
Maintaining and updating HOA with system usage data / analyzing user activity logs
Presenting Recommendations and Search result-list
User Search and Recommendations Lucent Backend
Adaptation Model
Twittomender Lucent ineraction Query Terms/ User terms
Twittomender Framework
Application Model
collaborative filtering (user tweets)
tweets content filtering
AHS Overlay Model
User Context
Usage Context
Context Model
Group Model
Followers/ Folowees
Retrieving group info from Twitter
Twitter API
Twittomender Profiler
User Model
User Information/ Service Request
Twitter API
Resource Model
Twitter
Retrieving info for Recommendations (R)
Lucene indexer
Lucene
Domain Model
Index
User choosing search or recom.
Twitter account
Search query
Goal Model

Fig. 3. Twittomender compliance with Generic Adaptation Framework

- Recommendation of resources of a different nature and type employs a separation of domain and resource models. Considering conceptual representation irrespectively of the resource type facilitates the usage of different information sources and resource types, mostly dealing with their conceptual representation. Thus not only different types of information resources (e. g. text, images, tweets, audio, video, etc.) but also heterogeneous resources (e. g. news, archives, e-learning repositories, online-shops, etc.) can be recommended within a single framework based on the conceptual representation of a specific domain. On the other hand considering resource and domain (conceptual structure) models separately takes a step towards solving the problem of the universal recommendations bridging the semantic gap. In this respect we have concept space and feature space with rankings that serves as a basis for transparent personalization irrespectively of the content type. This is perfectly shown in *Twittomender* where we consider tweets as a content base (resources) to provide user recommendations.
- User modeling — the GAF UM consists of entities (or essentially concepts if we consider an overlay model) for which we store a number of attribute-value pairs. For each of these entities there may be different attributes, representing various aspects of user profile. It implies using both short- and long-term user preferences and implements a great variety of user preferences such as users' tastes, interests, needs. In order to provide independence and flexibility of goals selection we distinguish goal model, which essentially can be used to recommend certain goals to follow based on the user preferences.
- Contextualization — as in AHS context awareness will help to decouple and make AH and RS and applications less integrated with the environment. On the other hand, considering a context model will allow the system to be sensitive and adapt in many other ways, rather than following a certain number of fixed adaptation rules or recommendation patterns [1]. Thus we devise a separate Context Model which might be an overlay of DM (and UM correspondingly) and represent both usage context (additional properties defining how a particular item/concept from a domain model should be used, under which conditions) or user context (e. g. certain items can be shown or recommended only in a particular context or each item is augmented with the additional contextual explanation with helps to make recommendation list more trustworthy).

6 Conclusions, Lessons Learned and Future Work

The coming years will bring more use-cases of how AHS can provide adaptation and recommendation, what techniques will be introduced, and what research areas will introduce new technologies in its evolution. So far a study of existing approaches in recommender systems was done to comply with the layered structure of adaptive information systems, which has resulted in an overlay presented in Fig. 3 providing an overview of a Twitter-based recommender systems and a corresponding overlay of AHS layers and adaptation process.

In this paper we investigated a general-purpose AHS architecture , which brings us new challenges to investigate the applicability of different recommendation approaches, as well as new developments in adaptive information systems. However, as a result of investigation now we can foresee some further developments and research strategies of bringing recommendations to the field of AHS and thus try to come up with up-to-date requirements for a modular composition of a GAF reference model which would be able to serve as well as a start-up for the Recommender system development using heterogeneous information sources. At the same time case-study helped to identify possible *Twittomender* improvements and extensions.

As part of future work we plan to make some further developments to *Twittomender*, one avenue which we are exploring is a mechanism to focus on users individual personal traits. What topics do user's talk about? What types of people do they follow? We plan to extend the *Twittomender* platform to cluster similar users based on these traits. This will allow Twitter users to quickly navigate to the types of people they would normally tend to gravitate towards or conversely show them the topics they would be clustered into e.g Sports, Technology, etc. As part of continued qualitative tests of *Twittomender*, we plan to extend our user trials. One test that will be carried out against Twitter's own recommendation system, this trial will check users satisfaction with the recommendation of both systems.

7 Acknowledgements

This work has been supported by the *NWO GAF: Generic Adaptation Framework* project, by *Science Foundation Ireland* under grant 07/CE/I1147 and by *Amdocs Inc.*.

References

1. G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer, 2011.
2. P. Brusilovsky, I. Cantador, Y. Koren, T. Kuflik, and M. Weimer. Workshop on information heterogeneity and fusion in recommender systems (hetrec 2010). In X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, editors, *RecSys*, pages 375–376. ACM, 2010.
3. S. Garcia Esparza, M. P. O'Mahony, and B. Smyth. On the real-time web as a source of recommendation knowledge. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 305–308, New York, NY, USA, 2010. ACM.
4. J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 199–206, New York, NY, USA, 2010. ACM.
5. E. Knutov, P. D. Bra, and M. Pechenizkiy. Generic adaptation framework: a process-oriented perspective. *J. Digit. Inf.*, 12(1), 2011.
6. Y. Koren and R. M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.
7. O. Phelan, K. McCarthy, and B. Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 385–388, New York, NY, USA, 2009. ACM.
8. F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.
9. J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: a comparison of microblog search and web search. In I. King, W. Nejdl, and H. Li, editors, *WSDM*, pages 35–44. ACM, 2011.

Towards a Framework for Adaptive Faceted Search on Twitter

Ilknur Celik[1], Fabian Abel[1], Patrick Siehndel[2]

[1] Web Information Systems, Delft University of Technology
{celik,abel}@tudelft.nl
[2] L3S Research Center, Leibniz University Hannover, Germany
siehndel@l3s.de

Abstract. In the last few years, Twitter has become a powerful tool for publishing and discussing information. Yet, content exploration in Twitter requires substantial efforts and users often have to scan information streams by hand. In this paper, we approach this problem by means of faceted search. We propose strategies for inferring facets and facet values on Twitter by enriching the semantics of individual Twitter messages and present different methods, including personalized and context-adaptive methods, for making faceted search on Twitter more effective. We conduct a preliminary analysis that shows that semantic enrichment of tweets is essential for faceted search on Twitter and that there is essential need for adaptive faceted search on Twitter. Furthermore, we propose an evaluation methodology that allows us to automatically evaluate the quality of adaptive faceted search on Twitter without requiring expensive user studies.

Key words: faceted search, twitter, semantic enrichment, adaptation

1 Introduction

With the growing information space on the Web and the increasing popularity of Social Media, Social Web applications became part of daily activities as well as the source of information for millions of people. The dynamic nature of the Web and the diversity of the users along with the heavy information load demanded some form of adaptation or personalization in many Web-based applications in various domains. Nowadays, many Social Web applications are suffering from similar information overload problems, where the users of these applications find it difficult to read, find and follow the relevant and interesting information shared by a large network of other users. Our research focuses on tackling information overload in one of the most popular of these applications, Twitter.

Twitter is the most popular micro-blogging site and a growing Social Web phenomenon that is attracting interest from different types of people all around the world for a variety of different purposes, such as fast communication, work, status updates, following news, sports, events, opinions, hot topics, and so on [1–8]. With millions of Twitter messages (tweets) per day, highly active users are estimated to receive hundreds of tweets every day[3]. Due to the lack of any adaptive or personalized navigation support in Twitter, users may get lost, become de-motivated and frustrated in this network of information overload [10]. Accessing required or interesting fresh content easily is vital in today's information age. Hence, there is a need for an effective personalized searching option from the users' point of view that would assist them in following the optimal path through a series of facets to find the information they are looking for, while providing a structured environment for relevant content exploring. Our research focuses on investigating ways to enhance searching and browsing in microblogging sites like Twitter by means of adaptive and personalized faceted search.

Searching and browsing are, indeed, somewhat limited in Twitter. For example, one can search for tweets by a keyword or by a user in a timeline that would return the most recent posts. So, if a user wants to see the different tweets about a field of sports, and were to search for "sports" in Twitter, only the recent tweets that contain the word "sports" would be listed to the user. Many tweets that do not contain the search keyword, but are about different sport events, sport games and sport news in general, would not be returned. Moreover, the Twitter keyword search differs from the general Web search due to the restricted message size of 140 characters in Twitter [9]. Traditional faceted search interfaces allow users to search for items by specifying queries regarding different dimensions and properties of the items (facets) [11]. For example, online stores such as eBay[4] or Amazon[5] enable narrowing down their users' search for products by specifying constraints regarding facets such as the price, the category or the producer of a product. In contrast, information on Twitter is rather unstructured and short, which does not explicitly feature facets. This puts constrains on the size and the number of keywords, as well as facets, that can be used as search parameters without risking to filter out many relevant results. Hence, searching by more than one topic (multiple facets), such as "sport events", would return only those recent tweets that contain both of these words and miss tweets like *"Off to BNP Paribas at Indian Wells"*, which mentions the name and the location of a sport event without necessarily including the keywords. In this paper, we introduce an adaptive faceted search framework for Twitter and investigate how to extract facets from tweets, how to design appropriate faceted search strategies on Twitter and how to evaluate such a framework. Our main contributions can be summarized as follows.

Semantic Enrichment We present methods for enriching the semantics of tweets by extracting facets (entities and topics) from tweets and related external Web resources.

User and Context Modeling Given the semantically enriched tweets, we propose user and context modeling strategies that identify (current) interests of a given Twitter user and allow for contextualizing the demands of this user.

[3] `http://techcrunch.com/2010/06/08/twitter-190-million-users/`
[4] `http://ebay.com/`
[5] `http://amazon.com/`

Adaptive Faceted Search We introduce faceted search strategies for content exploration on Twitter and propose methods that adapt to the interests and context of a user.

Evaluation Framework We present an evaluation environment based on simulated users to evaluate different strategies in our adaptive faceted search engine on Twitter.

2 Related Work and Our Motivation

The exponential growth of Twitter has attracted significant amount of research from various perspectives and fields recently. In this section, we focus on the related work that motivates and inspires our work, as well as relating our work to the existing literature.

2.1 Content Exploration on Twitter

A prototype for topic-based browsing in Twitter was proposed after observing how the users manage the incoming flood of updates [10]. This prototype interface, called Eddi, visualizes a user's Twitter feed using topic clusters constructed via a topic identification algorithm without using any semantics or natural language processing. This approach, however, does not find the relations between the topics or perform any recommendation of related topics. While it provides a means for browsing through a user's own feed by topics, our ambition is to infer relations between entities of all tweets in the network in order to adapt the list of facets presented to contain the related entities of the tweet of interest even outside of the user's feed. The aim is to provide a means where not only the users can easily reach to the information they are looking for by controlling their search parameters as they move along, but can also browse the related information about the current subject of interest by related people, countries, cities, events, and other selected facets.

2.2 Semantic Enrichment of Tweets

The main problem in searching microblogging platforms is the size of the messages. For example, the Twitter messages, with 140 characters limit, are too short to extract meaningful semantics on their own. Furthermore users tend to use abbreviations and short-form for words to save space, as well as colloquial expressions, which make it even harder to infer semantics from tweets. Rowe et al. mapped tweets to conference talks and exploited metadata of the corresponding research papers to enrich the semantics of tweets to better understand the semantics of the tweets published in conferences [12]. We follow a similar approach to this, except we try to enrich the tweets in general and not in a restricted domain like scientific conferences. A study by Kwak et al. revealed that the majority of the trending topics in Twitter are either headline or persistent news, with 85% of all the posted tweets being related to news, claiming Twitter is used more as a news media than a social network [4]. Consequently, we try to map tweets to news articles on the Web over the same time period in order to enrich them and to allow for extracting more entities to generate richer facets.

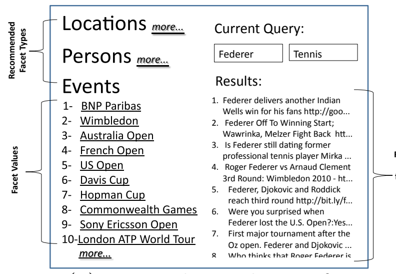
2.3 User and Context Modeling for Adaptive Faceted Search in Twitter

We also try to discover the relations between the extracted entities by studying different strategies in order to determine relatedness relations between entities such as persons related to an event and identify any temporal constraints on such relations. These learnt relations between entities can be utilized to ease the search by grouping together the related facets and recommending the most relevant facets that the user is looking for. Marinho et al. proposed a method for collabulary learning which takes a folksonomy and domain-expert ontology as input and performs semantic mapping to generate an enriched folksonomy [13]. An algorithm based on frequent itemsets techniques is then applied to learn an ontology over this enriched folksonomy. A similar approach exploited frequent itemsets to learn association rules from tagging activities [14]. We study the co-occurrence frequencies of entity pairs and compare these with other strategies for tweets in combination with news articles to learn relations between these entities.

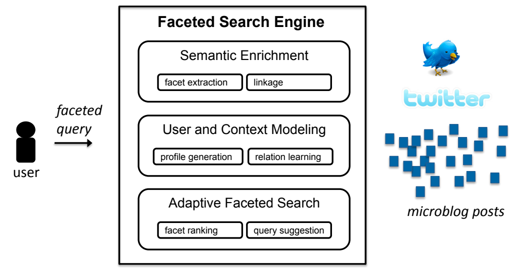
In addition to adapting the facets to the current search, we aim at adapting the facet values to the current state of the users in order to personalize the search and content exploration. Liu et al. analyzed content-based recommenders for Google News and showed that interests in news topics such as technology, politics, et cetera change over time [15]. They also predicted user interests and showed that these user profiles in combination with recent trends on Google News outperform collaborative filtering. Similarly, Chen et al. studied content recommendation in Twitter and found out that both topic and relevance are important considerations [16]. They also observed that URLs extracted from the user's close social group is more successful than the most popular ones. Correspondingly, we observe the users' past activities to infer their recent interests based on their recent tweets and re-tweets. In other words, we build a profile of user interests in accordance with entities and topics, which is then used to adapt ranking of the facet values. Re-arranging the facet values according to user history and interests in line with the trendy topics can accelerate and thus improve the searching experience.

3 Faceted Search on Twitter

On Twitter, facets describe properties of a Twitter message. For example, persons that are mentioned in a tweet or events a tweet refers to. Oren et al. [11] formulate the problem of faceted search in RDF terminology. Given an RDF statement ($subject, predicate, object$), the faceted search engine interprets (i) the subject as the actual resource that should be returned by the engine, (ii) the predicate as the facet type and (iii) the object as the facet value (restriction value). A faceted query (facet-value pair) that is sent to a faceted search engine thus consists of a predicate and an object. We follow this problem formulation proposed by Oren et al. [11] and interpret tweets as the actual resources the faceted search engine should return. If a tweet (subject) mentions an entity then


Recommended Facet Types
Locations more...
Persons more...
Events
Facet Values
1- BNP Paribas
2- Wimbledon
3- Australia Open
4- French Open
5- US Open
6- Davis Cup
7- Hopman Cup
8- Commonwealth Games
9- Sony Ericsson Open
10-London ATP World Tour
more...
Current Query:
Federer
Tennis
Results:
1. Federer delivers another Indian Wells win for his fans http://goo...
2. Federer Off To Winning Start; Wawrinka, Melzer Fight Back htt...
3. Is Federer still dating former professional tennis player Mirka ...
4. Roger Federer vs Arnaud Clement 3rd Round: Wimbledon 2010 - ht...
5. Federer, Djokovic and Roddick reach third round http://bit.ly/f...
6. Were you surprised when Federer lost the U.S. Open?:Yes...
7. First major tournament after the Oz open. Federer and Djokovic ...
8. Who thinks that Roger Federer is
Resulting tweets from the current query

(a) Faceted search interface



(b) Faceted search architecture

Fig. 1. Adaptive faceted search on Twitter: (a) example interface and (b) architecture of the faceted search engine.

the type of the entity is considered as facet type (predicate) and the actual identifier of the entity is considered as facet value (object). For example, given a tweet t that refers to the tennis player "Federer", the corresponding URI of the entity ($URI_{federer}$) and the URI of the entity type (URI_{person}) are used to describe the tweet by means of an RDF statement: $(t, URI_{person}, URI_{federer})$.

Figure 1(a) illustrates how we envision the corresponding faceted search interface that allows users to formulate faceted queries. Given a list of facet values which are grouped around facet types such as locations, persons and events, users can select facet-value pairs such as $(URI_{event}, URI_{wimbeldon})$ to refine their current query $((URI_{person}, URI_{federer}), (URI_{sportsgame}, URI_{tennis}))$. A faceted query thus may consist of several facet-value pairs. Only those tweets that match all facet-value constraints will be returned to the user. The ranking of the tweets that match a faceted query is a research problem of its own and could be solved by exploiting the popularity of tweets – e.g. measured via the number of re-tweets or via the popularity of the user who published the tweet (cf. [17]). The core challenge of the faceted search interface is to support the facet-value selection as good as possible. Hence, the facet-value pairs that are presented in the faceted search interface (see left in Figure 1(a)) have to be ranked so that users can quickly narrow down the search result lists until they find the tweets they are interested in. Therefore, the *facet ranking problem* can be defined as follows.

Definition 1 (Facet Ranking Problem). *Given the current query F_{query}, which is a set of facet-value pairs $(predicate, object) \in F_{query}$, the hit list H of resources that match the current query, a set of candidate facet-value pairs $(predicate, object) \in F$ and a user u, who is searching for a resource t via the faceted search interface, the core challenge of the faceted search engine is to rank the facet-value pairs F. Those pairs should appear at the top of the ranking that restrict the hit list H so that u can retrieve t with the least possible effort.*

The effort, which u has to invest to narrow down the search result list H, can be measured by click and scroll operations. Strategies for facet ranking are discussed in Section 3.2.

3.1 Architecture for Adaptive Faceted Search on Twitter

Figure 1(b) illustrates the architecture of the engine that we propose for faceted search on Twitter. The main components of the engine are the following.

Semantic Enrichment The semantic enrichment layer aims to extract facets from tweets and generate RDF statements that describe the facet-value pairs which are associated with a Twitter message. In particular, each tweet is processed to identify entities (facet values) that are mentioned in the message. We therefore make use of the OpenCalais API[6], which allows for the extraction of 39 different types of entities (facet types) including persons, organizations, countries, cities and events. As Twitter messages are limited to 140 characters, the extraction of entities from tweets is a non-trivial problem. Thus, we introduced a set of strategies that link tweets with external Web resources (news articles) and propagate the semantics extracted from these resources to the related tweets in [18]. For example, given a tweet “This is great http://bit.ly/2fRds1t”, we extract entities from the referenced resource (http://bit.ly/2fRds1t) and attach the extracted entities to the tweet. In our analysis, we show that this semantic enrichment allows us to significantly better prepare the tweets for faceted search than enrichment which is merely based on tweets.

User and Context Modeling In order to adapt the facet ranking to the people who are using the faceted search engine, we propose user modeling and context modeling strategies. The user modeling strategies model the interests of the users in certain facet values (entities and topics). We therefore exploit the tweets that have been published (including re-tweets) by a user. In future work, we also plan to consider click-through data from the faceted search engine. Context modeling covers mining of new knowledge from the Twitter data. We therefore propose relation learning strategies that exploit co-occurrence of entities in Twitter messages to infer typed relationships between entities [19].

Adaptive Faceted Search Based on the semantically enriched tweets, the learnt relationships between entities extracted from tweets and the user profiles generated by the user modeling layer, the adaptive faceted search layer solves the actual facet ranking problem. It provides methods that adapt the facet-value pair ranking to the given context and user. Furthermore, it provides query suggestions by exploiting the relations learnt from the Twitter messages. Given the current facet query, which is a list of facet-value pairs where each value refers to an entity, we can exploit relationships between entities in order to identify entities that are related to those entities that occur in the current facet query. We leave the analysis of such query suggestions for future work. Instead, we focus on the facet ranking problem and propose different strategies for ranking facet-value pairs in the next subsection.

[6] `http://www.opencalais.com/`

3.2 Adaptive Faceted Search and Facet Ranking Strategies

Non-Personalized Facet Ranking A lightweight approach is to rank the facet-value pairs $(p, e) \in F$ based on their occurrence frequency in the current hit list H, the set of tweets that match the current query (cf. Definition 1):

$$rank_{frequency}((p, e), H) = |H_{(p,e)}| \tag{1}$$

$|H_{(p,e)}|$ is the number of (remaining) tweets that contain the facet-value pair (p, e) that can be applied to further filter the given hit list H. By ranking those facets that appear in most of the tweets, $rank_{frequency}$ minimizes the risk of filtering out relevant tweets but might increase the effort a user has to invest to narrow down search results.

Context-adaptive Facet Ranking The context-adaptive strategy exploits relationships between entities (facet values) to produce the facet ranking. A relationship is therefore defined as follows:

Definition 2 (Relationship). *Given two entities e_1 and e_2, a* relationship *between these entities is described via a tuple $rel(e_1, e_2, type, t_{start}, t_{end}, w)$, where* type *labels the relationship, t_{start} and t_{end} specify the temporal validity of the relationship and $w \in [0..1]$ is a weighting score that allows for specifying the strength of the relationship.*

The higher the weighting score w the stronger the relationship between e_1 and e_2. We use co-occurrence frequency as weighting scheme. Hence, given the enriched tweets, we count the number of tweets both entities (e_1 and e_2) are associated with. The context-adaptive facet ranking strategy ranks the facet-value pairs $(p, e) \in F$ according to $w(e_i, e)$, where e_i is a facet value that is already part of the given query: $(p_i, e_i) \in F_{query}$ (cf. Definition 1):

$$rank_{relation}((p, e), F_{query}) = \sum_i w(e_i, e)|(p, e_i) \in F_{query} \tag{2}$$

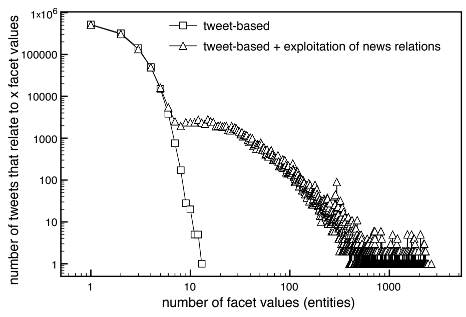
Hence, the context-sensitive strategy can only be applied in situations where the user has already made one selection, so that $|F_{query}| > 0$.

Personalized Facet Ranking The personalized facet ranking strategy adapts the facet ranking to a given user profile that is generated by the user modeling layer depicted in Figure 1(b). User profiles conform to the following model and specify a user's interest into a specific facet value (entity).

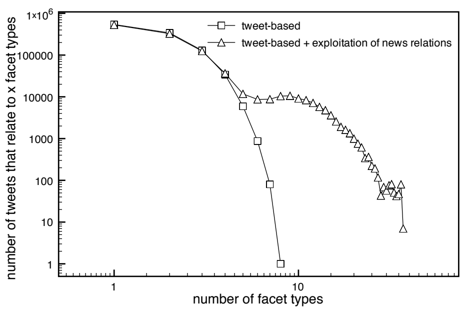
Definition 3 (User Profile). *The profile of a user $u \in U$ is a set of weighted entities where with respect to the given user u for an entity $e \in E$ its weight $w(u, e)$ is computed by a certain function w.*

$$P(u) = \{(e, w(u, e))|e \in E, u \in U\}$$

Here, E and U denote the set of entities and users respectively.



(a) number of facet values per tweet



(b) number of facet types per tweet

Fig. 2. Impact of semantic enrichment on (a) the number of facet values per tweet and (b) the number of distinct facet types per tweet.

Given the set of facet-value pairs $(p, e) \in F$ (see Definition 1), the personalized facet ranking strategy utilizes the weight $w(u, e)$ in $P(u)$ to rank the facet-value pairs:

$$rank_{personalized}((p,e), P(u)) = \begin{cases} w(u,e) & \text{if } w(u,e) \in P(u) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

By combining the above three strategies it is possible to generate further facet ranking methods. A combination of two strategies can be realized by building the weighted average computed for a given facet-value pair (p, e) (e.g. $rank_{combined} = \alpha \cdot rank_{\alpha}((p, e)) + \beta \cdot rank_{\beta}((p, e))$).

4 Analysis of Faceted Search on Twitter

In our analysis, we study the characteristics of facets on Twitter. As described above, tweets do not feature many facets by nature. Therefore, strategies that enrich the semantic of tweets are required in order to derive facet-value pairs for tweets. In this section, we examine how the semantic enrichment supports the derivation of facets. Furthermore, we analyze the feasibility of the user and context modeling strategies for making faceted search on Twitter adaptive.

4.1 Analysis of Semantic Enrichment

As tweets do not provide facets related to the topic, our faceted search framework provides the functionality to enrich the semantics of tweets. To analyze the feasibility of our semantic enrichment component (see Section 3), we monitored the Twitter activities of more than 20,000 users over a period of more than two months and processed the data that we collected (1,671,389 tweets in total) to extract facet values from the tweets. For 62.91% of the tweets, we succeeded in extracting at least one entity that we can use as facet value. By making use of the semantic enrichment functionality that exploits links to external Web resources (and news articles in particular), we increased the coverage so that 66.77% of

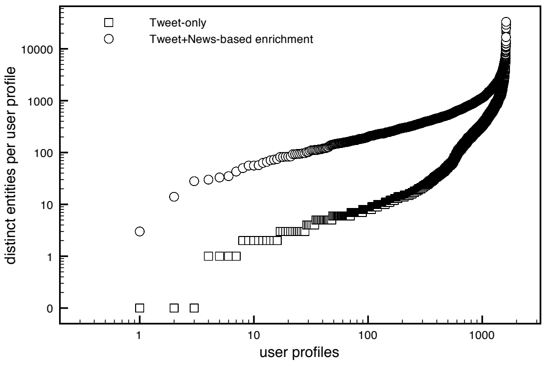


Fig. 3. Entity-based user profiles that can be exploited for personalized facet ranking.

the tweets which are enriched with facet values obtained from related news have at least one facet value. In the context of the news-based enrichment, we connected 458,566 Twitter messages with news articles of which 98,189 relations were explicitly given in the tweets by URLs that pointed to the corresponding news article. The remaining 360,377 relations were obtained by comparing the entities that were mentioned in both news articles and tweets as well as comparing the timestamps. In previous work we showed that this method correlates news and tweets with an accuracy of more than 70% [20].

Figure 2(a) reveals that the number of facet values increases clearly when tweets are enriched with entities of related news articles. For example, less than 20 tweets exhibit more than 10 facet values in the case of semantic enrichment that is merely based on tweets . Given that tweets are limited to 140 characters, this observation is expected. Moreover, the number of different facet types per tweet also increases when linkage to news articles is exploited (see Figure 2(b)). In our current implementation, we differentiate between 39 different facet types, where persons, countries and organizations are the most popular types of facets. In Figure 2(b), we see that the tweet-based enrichment does not allow for more than 10 different types of facet types per tweets while the exploitation of news relations features more than 10,000 tweets that can be discovered via more than 10 different facet types, i.e. users can choose between various facets to narrow down the actual hit list (cf. Figure 1(a)).

4.2 Analysis of User and Context Modeling

The adaptation of the faceted search interface to the preferences of the user and therefore the personalized facet ranking strategy (see Equation 3) requires entity-based user profiles (see Definition 3). To analyze to what extent this method can succeed, we show the profile size of 1500 randomly selected user profiles in Figure 3. We see that the news-based enrichment results in profiles that provide more entities than the tweet-only based enrichment. For example, semantic enrichment based merely on tweets fails for three users as the size of the profile is zero for these users. In contrast, the news-based enrichment successfully generates profiles for all users. For more than 98% of the users, the number of distinct entities per profile is even higher than 100. This indicates that news-based enrichment prevents from sparsity problems and thus allows for supporting the personalized facet ranking better than the tweets-only-based enrichment.

5 Evaluation Framework for Faceted Search

Evaluating the performance of faceted search is challenging. It usually requires query logs and click-through data, which is difficult to get for researchers, or calls for user studies, which are expensive if they are conducted on a large scale. In this section, we propose a novel technique for automatically evaluating the performance of faceted search on Twitter. Our evaluation methodology follows an idea introduced by Koren et al. [21] and exploits re-tweets as ground truth for estimating user relevance. The evaluation methodology is based on simulated users who behave in a predefined way. The utility of the interface is measured by the actions a simulated user needs to perform in order to find a relevant document.

General Setup. The general setup used for the evaluation process contains parameters describing the user interface itself and algorithms characterizing the simulated user behavior. In general, all faceted search user interfaces share some common characteristics and contains at least two parts: an area displaying the facets and a part showing the search results. For our evaluation process, the number of documents to be presented at a time, the number of different facets to be displayed and the number of elements which can be shown for each different facet need to be defined. We setup a basic framework for a search interface by defining these three parameters. Based on this interface, a user can perform different actions, where the goal is to find a relevant document. For every action we can define a cost, where the cost is related to the time a real user would need to accomplish this action. In our scenario a user can perform the following actions:

Select facet-value pair Basic action a user performs every time a facet-value pair is clicked, where the displayed search results are automatically updated after the selection (costs: 1).

View more facet-value pairs This action indicates that none of the currently displayed facet-value pairs are relevant for the user. By performing this action the user gets an additional amount of facet-value pairs related to one facet (costs: 2).

Show more documents This action allows the user to see more documents (tweets) matching the currently selected facet-values (costs: 2).

Select relevant tweet This action ends the current search (costs: 0).

Beside the actions mentioned above one could also consider the act of deselecting previously marked facet-values. In our search scenario, this action is not included as we assume that the users have perfect knowledge about the tweet they are looking for, and therefore a wrong selection will not take place.

Selection Strategies. The simulated users select facet-value pairs based on different strategies. The strategies we use for our evaluation are:

Random user This user randomly selects one of the displayed facet-values which matches the tweet he is looking for. If none of the displayed facet-value pairs matches the tweet, he randomly chooses one facet to see more facet-value pairs.

First-match user This user selects the first matching facet-value pair displayed by the interface. The basic idea behind this strategy is based on a user who directly clicks on a matching facet-value pair suggestion and do not look at all displayed facet value pairs to find the best matching one.

Greedy user This strategy tries to reduce the number of matching documents as fast as possible. This user selects the facet-value pair which occurs in the least number of remaining documents. This can be motivated by a user who selects the facet-value pair which is particularly important for the targeted tweet, in comparison to facet-value pairs which are related to many tweets.

Based on these facet selection strategies, the simulated user searches for a relevant document. The cost of this search is measured by the costs and number of actions a user needs to perform to find a relevant document.

Evaluation process. To measure the benefit of the proposed methods for faceted search, we evaluate the cost for a user to find relevant documents. Here, a tweet is relevant to a user, if the user re-tweeted this tweet. Re-tweeting a tweet indicates that the user has read the tweet and is to some extend interested in the content of the tweet. The proposed method is used to compare the costs of finding a relevant document when using the baseline ranking strategy based on frequency (non-personalized facet ranking) in comparison with context-adaptive facet ranking and personalized facet ranking.

6 Conclusions

In this paper, we presented an adaptive and personalized faceted search engine for Twitter, where we explained approaches for enriching the semantics of tweets, extracting facets, discovering relatedness information between entities and observing user activities to learn their behavior and interests in order to support users in their search for specific information or tweets. We proposed different strategies based on learnt relations together with user action history for adapting the search behavior as well as improving content exploration in Twitter. Furthermore, we introduced a generic evaluation environment based on Koren et al. [21] that will allow us to evaluate our strategies by simulated experiments, which constitutes part of our future research.

Acknowledgements The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no ICT 257831 (ImREAL project[7]).

[7] `http://imreal-project.eu`

References

1. Hughes, A.L., Palen, L.: Twitter Adoption and Use in Mass Convergence and Emergency Events. In: Proc. of ISCRAM. (2009)
2. Zhao, D., Rosson, M.B.: How and why people Twitter: the role that micro-blogging plays in informal communication at work. In: Proc. GROUP, ACM (2009) 243–252
3. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring User Influence in Twitter: The Million Follower Fallacy. In: Proc. of ICWSM, The AAAI Press (2010)
4. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proc. of WWW, ACM (2010) 591–600
5. Lerman, K., Ghosh, R.: Information contagion: an empirical study of spread of news on digg and twitter social networks. In: Proc. of ICWSM, The AAAI Press (2010)
6. Java, A., Song, X., Finin, T., Tseng, B.: Why we twitter: understanding microblogging usage and communities. In: Proc. of WebKDD/SNA-KDD, ACM (2007) 56–65
7. Kaufman, S.J., Chen, J.: Where we Twitter. In: Proc. of Workshop on Microblogging: What and How Can We Learn From It? (2010)
8. Romero, D.M., Meeder, B., Kleinberg, J.: Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter. In: Proc. of WWW, ACM (2011)
9. Teevan, J., Ramage, D., Morris,M.R.: #TwitterSearch: A Comparison of Microblog Search and Web Search. In: Proc. of WSDM, ACM (2011)
10. Bernstein, M., Kairam, S., Suh, B., Hong, L., Chi, E.H.: A torrent of tweets: managing information overload in online social streams. In: Proc. of Workshop on Microblogging: What and How Can We Learn From It? (2010)
11. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for rdf data. In: Proc. of ISWC, Springer (2006) 559–572
12. Rowe, M, Stankovic, M., Laublet, P.: Mapping Tweets to Conference Talks: A Goldmine for Semantics. In: Proc. of SDoW, colocated with ISWC, CEUR-WS.org (2010)
13. Balby Marinho, L., Buza, K., Schmidt-Thieme, L.: Folksonomy-based collabulary learning. In: Proc. of ISWC, Springer (2008) 261–276
14. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Emergent Semantics in BibSonomy. In: Informatik für Menschen. Volume 94(2) of LNI, GI (2006)
15. Liu, J., Dolan, P., Pedersen, E.R.: Personalized news recommendation based on click behavior. In: Proc. of IUI, ACM (2010) 31–40
16. Chen, J., Nairn, R., Nelson, L., Bernstein, M., Chi, E.: Short and tweet: experiments on recommending content from information streams. In: Proc. of CHI, ACM (2010) 1185–1194
17. Weng, J., Lim, E.P., Jiang, J., He, Q.: Twitterrank: finding topic-sensitive influential twitterers. In: Proc. of WSDM, ACM (2010) 261–270
18. Abel, F., Gao, Q., Houben, G.J., Tao, K.: Analyzing User Modeling on Twitter for Personalized News Recommendations. In: Proc. of UMAP, Springer (2011)
19. Celik, I., Abel, F.: Learning Semantic Relationships between Entities in Twitter. In: Proc. of ICWE, (2011)
20. Abel, F., Gao, Q., Houben, G.J., Tao, K.: Semantic Enrichment of Twitter Posts for User Profile Construction on the Social Web. In: ESWC, Springer (2011)
21. Koren, J., Zhang, Y., Liu, X.: Personalized interactive faceted search. In: Proc. of WWW, ACM (2008) 477–486

Referencing within evolving hypertext

Victor Grishchenko, Janus A. Pouwelse, and Henk Sips

Delft University of Technology
Mekelweg 4, 2628CD
Delft, The Netherlands
`victor.grishchenko@gmail.com`

Abstract. The classic hypertext model omits the process of text growth, evolution and synthesis. With hypertext creation becoming increasingly collaborative and change timescales becoming shorter, explicitly addressing text evolution is the key to the next stage of hypertext development. Uniform Resource Identifier (URI) is a proven general concept that enabled the Web. In application to versioned *deep* hypertext, expressive power of a classical hyperlink becomes insufficient.
Based on the Causal Trees model, we introduce a minimalistic but powerful query language of *specifiers* that provides us great flexibility of referencing within a changing hypertext. Specifiers capture the state of the text, point at changes, expose authorship or blend branches. Being a part of an URI, a specifier puts advanced distributed revision control techniques within reach of a regular web user.

1 Introduction

In the WWW/HTML model and, generally, in "chunked hypertext" systems the main addressable unit is a "page" which might optionally also have addressable "anchors" inside it. That is generally sufficient as long as we deal with static texts, albeit the requirement that a page author must pre-provision anchors is limiting. However, if we follow the general vision of a text as an evolving entity (the "wiki model"), then the expressive power of a classical hyperlink is insufficient. Since the addressed text is continuously changing, anchors might disappear, and the content that is actually addressed by the link might be reedited or its surroundings may change. Similarly, there is no standard way of pointing at particular statements and passages in the text. For collaboratively created texts, such a possibility is desirable. Also, there is no semantics in place to address co-existing versions of a text (named "branches" in the version control parlance). Those might be drafts, reeditions, alternative versions. Thus, our mission is to explore possible approaches and conventions of referencing particular parts of text, its particular versions, or both. We want to measure, mark and cut text in breadth and depth!

This paper is structured as follows. First, we consider relevant existing models and their limitations in Section 2. Section 3 briefly describes the Causal Trees (*ct*) model of text versioning and the basic primitives available for text/ version addressing. In Section 4, based on the URI specification, we define the syntax of specifiers.In Section 5 we consider practical applications for the proposed conventions, explained as simple Alice-Bob scenarios. The Section 6 concludes.

2 Related and previous work

The early hypertext system Xanadu employed Dewey-inspired change-resistant addresses named tumblers, e.g. `1.2368.792.6.0.6974.383.1988.352.0.75.2.0.1.9287` (an example from [21] addressing a particular point in a particular version of a document). That scheme was not reused by any later system.

Today, most wikis, including Wikipedia, have a history view capable of retrieving and comparing different versions of a text. However, the URL syntax is implementation-dependent. The authors are unaware of any wiki that allows for branching/ multiversioning of texts; a document's history is always seen as a linear sequence of numbered revisions. Distributed revision control systems[1] implement an extensive toolset for identifying/processing parallel revisions of texts (typically, source code). Most of those systems model a mutation history as a directed acyclic graph of revisions. The inner contents of a file are considered a single data piece; no fine-grained addressing is possible. Revisions are typically identified by cryptographic hashes of the content and metadata. A number of wikis[2] use distributed version control systems as their back-end, but they don't pass on the branching functionality to the front-end.

The possibility of addressing precise parts of a text is a well-known general problem. Texts that need repeated reading, referencing or modification typically have some fragment addressing scheme as well. Examples are Biblical (e.g. `1 Kings 11:41`), Qur'an (`2:2`), legal (`U.S.Const.am.8.`) or source code (`kernel/panic.c:57`) references. Paper books are referenced using page numbers, but those might change from edition to edition. These days, various e-book devices made the notion of "a page" completely ephemeral. In application to computer hypertext, three classic examples of addressing schemes are Purple numbers [16], XPointer [7] and the classic `patch` [6] format. They rely on three basic techniques: offsets, anchors and/or context. The first and the simplest addressing technique is to use word/symbol offsets within a file. That works well for static files. For example, web search engines employ inverted indexes that list all the document-offset pairs where a particular word was found. But, in a changing text, new edits invalidate offsets. Hence, every next version has to be processed as a separate text. The second technique is planting anchors within the text. However, pre-provisioning anchors for any future use by any third party is not practical. The third technique is to address a point in the text by mentioning its *context*, i.e. snippets of surrounding text. The approach is robust, but heavyweight, dependent on heuristics and also vulnerable to text mutations. Combining those techniques may increase robustness, e.g. the UNIX

[1] For example, Bazaar http://bazaar.canonical.com/, Git http://git-scm.com/, Mercurial http://mercurial.selenic.com/

[2] For example, Gitit wiki http://gitit.net or git-wiki http://github.com/sr/git-wiki

`diff/patch` format employs approximate offsets *and* context snippets. But, that might increase fragility as well; e.g. an XPointer relying on an anchor *and* an offset becomes vulnerable to changes in both. Purple numbers address paragraphs using either offsets or anchors. Neither method is perfect.

Several well-known technologies, such as WebDAV [13], BigTable [10] or Memento [23], represent history of an evolving Web page as a sequence (or a graph) of revisions identified by either timestamps or arbitrary labels. In that model, every version stays a separate monolithic piece. There are some efforts to apply versioning to adaptive hypertext [17].

The Operational Transformation theory (OT, [11]) generalized offset-based addressing scheme for changing texts with the purpose of real-time revision control in distributed systems. Currently OT is employed by Google Docs, Google Wave and other projects[3]. Among the shortcomings of the OT theory is its high complexity and long-standing correctness issues [19, 15]. Systems that are known to work had to adopt compromises on the original problem statement [11], either by relying on a central coordinating entity [3] or by requiring that edits are always merged in the same exact order [22]. Still, the main problem is bigger: OT does not address revision-control tasks that lie outside the frame of real-time collaborative editing, narrowly defined. Those are: branching, merging, propagation of changes, "blame maps", diffs and others.

The Causal Trees (*ct*) model [14] was introduced to resolve the problems evident in OT. Instead of relying on offset-based addressing, which is volatile once we consider a changing text, *ct* assigns unique identifiers to all symbols of the text. Thus, it trivially resolves the correctness/complexity problems and also introduces new possibilities. For example, it allows fine-grained fragment addressing that survives edits. Being defined along the lines of the Lamport-Fidge [18, 12] time/event model, it allows for reliable identification of any versions, even in a text that has multiple concurrently changing editions (branches). Effectively, *ct* implements the functionality of deep hypertext, as described in [1]. The *ct* model is the starting point of this work. Recently, the *ct* model was implemented as a JavaScript library *ctre.js*[4].

Consider a document which has several evolving branches. Suppose it is a Wikipedia-style wiki of course materials which is supported by several universities in parallel. On the one hand, we want to maintain the upside of collaboration which is well illustrated by the success of Wikipedia. On the other hand, we want to avoid edit warring [2] and the extreme volatility of content typical of Wikipedia. Thus, we suppose that such a wiki is supported by a network of collaborators exchanging, negotiating and filtering edits in the way the Linux kernel is developed (the "git model").

In such an environment, a single document may be seen in hundreds of ways, depending on which editions we are looking at, and when. Similarly, if we want to address some particular parts of a document, there are numerous possibilities.

[3] Google Docs http://docs.google.com, Google Wave http://wave.google.com, Gobby http://gobby.0x539.de, Etherpad http://etherpad.org

[4] Project page at GitHub: http://github.com/gritzko/ctre

Our intention is to extend the semantics of URIs [8] to deal with that complexity and to make it manageable and clear to a regular user. We assume that the address field of the browser is the user's primary means of navigation. Finally, the ability to identify and address (and instantly access) arbitrary resources is the cornerstone of the Web. We extend that ability in space and time.

3 The ct model

The *ct* model augments the very fabric of text to reflect its evolution. Drawing some lessons from the history of Operational Transformation theory, *ct* does not use offset-based addressing and does not try to find one true frame of reference. Instead, *ct* closely follows the lines of the Lamport-Fidge [18, 12] relativistic model of events and time in a distributed system. In essence, *ct* is a Minkowski *spacetime* [20] model for versioned texts, unifying time (versions) and space (text) as different projections of the same phenomenon. This section will briefly explain the basics of the model and its building blocks.

A. Symbols have own identity. Attempts to identify symbols by their offsets in a versioned text have produced unsatisfactory results. As a text constantly changes, so do offsets. Thus, it becomes nearly impossible to reliably *point at* a given character. Instead, *ct* starts by assigning unique identifiers to every symbol in a document. Securing globally consistent serial identifiers is impossible in a truly decentralized system, thus we resort to the Lamport-Fidge approach. For a given document, all its symbols originating from the same author are sequentially numbered. Thus, they constitute a vector of contributions (called a *yarn*) of that particular author to that particular document. Still, we do not try to impose any global numbering. Instead, we identify a symbol by its (yarn_uri, symbol_number) pair, i.e. `(alice.org/page,398)`. Given that id, we may always retrieve the symbol, and a symbol may be reliably pointed at, independently of any changes in the document. A symbol with an identity is called an *atom*.

B. Text and operations are the same. There is no separation of "a text" and text-modifying "operations". A text consists of *atoms* and any operation is a set of atoms as well. An "atom" is a symbol plus its metadata. Even deletions are implemented as special "backspace" meta-symbols. An atom's metadata consists of its own identifier and an identifier of the *causing* atom. The causality relation weaves atoms together to form a text. Very much in the spirit of the Markov chain [9] model, a symbol is said to be *caused* by its preceding symbol at the time of insertion. Such a simple relation leads to provable correctness and convergence even in a distributed system with no central entities [14]. All replicas of a text eventually converge to the same state and no edits are misapplied.

C. All frames of reference are equal. A frame of reference corresponds to a single "local" author and his version of the text. There is no special "central", "reference" or "true" version. When accessing a text, we access not the text per se, but its version by a particular editor (a yarn). Other yarns are retrieved by recursively following causal dependencies. Then, yarns are woven together to produce a version of the text [14]. A transitive closure of causal dependencies is one of the key concepts of *ct*. For example, it defines the way branches are represented in *ct*. A branch is a set of yarns that has dependencies on the "trunk" yarns of the text, but there are no dependencies in the reverse direction (trunk to branch). Once such reverse dependencies are created, the branch effectively merges into the trunk, as it becomes a part of the trunk's closure.

3.1 Unicode serialization

In practice, *ct* is implemented with regular expressions[5]. That is not only mathematically well-defined, but also practically useful, as it allows to run *ct* in a Web browser with native speed. To make atoms regex-processable, their ids are encoded with two Unicode symbols, one for the author/yarn and another for the symbol's serial number. For example, an atom id `(alice.org/page,398)` becomes "AΘ", where "Θ" is the Greek capital letter Theta corresponding to Unicode code point 398. We also assume a mapping between symbols and URIs, where "A" corresponds to `alice.org/page`.

While two-symbol ids might seem insufficient, they allow for up to 4 billion symbols per document if using a baseline regex implementation supporting only 16-bit Unicode BMP characters [5]. Once an author exceeds the 2^{16} symbol limit, (s)he might be allocated another yarn id. The case of 2^{16} authors per document is considered highly unlikely, and even if that happens, there is always an option to use two characters for a yarn id (hence three characters per atom id total). Still, we believe that the two-symbol scheme provides sufficient numbering capacity for most of the texts.

Serializing atoms as tuples of Unicode symbols allows to pack all data structures into strings and to process them with regular expressions. That resolves an important practical bottleneck. Performing sophisticated revision control operations in a Web browser, in real time, becomes possible.

3.2 Specifying ranges and versions

In full accordance with the *spacetime* concept, *ct* denotes intervals in time (versions) and intervals in space (text ranges) in a very similar way, namely by mentioning their bounding atoms. Regarding text ranges, we may rely on the linear order of symbols in a text, and simply denote a text range by mentioning its end-points. Thus, $[A4; B8)$ stands for an interval starting at a symbol number 4 by author A and lasting till, but not including, the symbol number 8 by B.[6] This interval specification is immune to any further text changes, including deletion of the bounding atoms.

[5] The PCRE (Perl Compatible Regular Expressions) dialect, as used in JavaScript

[6] Parentheses () stand for excluded endpoints, square brackets [] for included.

Denoting versions in a distributed system may be trickier. In the simplest case, a version history is linear (e.g. there is only one author). Then, a revision may be denoted just by mentioning its most recently introduced symbol, e.g. $B8$. Thus, that one and all the "older" symbols $B1$–$B7$ constitute a version. But, in the general case of a distributed system, all participants are free to introduce new changes, and those changes propagate with finite speed. At a given moment in time, each participant sees a version of a text, based on the edits it is aware of. Thus, version history is not sequential. The only appropriate model is a directed acyclic graph. In such a case, a version might have multiple "most recent" symbols, e.g. $A4$ and $B8$. The number of such symbols cannot exceed the number of authors or, more precisely, yarns. Essentially, a set of those "most recent" symbols is a logical vector timestamp [18]. Under the hood, *ct* represents vector timestamps as *wefts*, which are strings of even length consisting of two-symbol atom identifiers, like `A4B8`.

Note that range/version specifications may have any of their bounding symbols either excluded or included. In general, that time-space unification will help us a lot with our specifier syntax (see Sec. 4.2).

4 URL conventions

An atom identifier is just a pair of Unicode symbols, one for the author/yarn code and another for the atom serial number within the yarn. A version or a text range is thus denoted by several Unicode symbol pairs. We want to use them in URIs as basic building blocks of our version/fragment specifiers. But, the URI syntax [8] only allows for a restricted subset of 7-bit ASCII symbols. So, we have to define a serialization of atom identifiers. Our end goal is to develop a syntax convention that allows to denote text versions and ranges by URIs that are easily communicated using email, instant messaging, Twitter, napkins [8] and even spoken speech.

4.1 Encoding

The default option for using Unicode in URIs is the percent-encoding [8]. But that would consume six characters per every non-ASCII symbol (hence, 12 per atom id). Instead, we encode atom identifiers using base64 encoding, namely its variety that employs alphanumerics, tilde and underscore to express $64 = 2^6$ values.[7] We only consider Unicode characters of the Basic Multilingual Plane, which gives us 2^{16} possible values for either author or symbol code, and correspondingly 2^{32} possible values of an atom id. We resort to separate encoding of author and symbol codes by up to three base64 characters each ($2^{6\times3} > 2^{16}$). Thus, an encoded atom id may take *up to* six base64 characters.

We expect most texts to be short, created by a few authors, so most author/symbol codes will have low values. As one URI may include many atom

[7] `0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz~_`

ids, it is highly beneficial to use a variable-length encoding to shorten serialized atom ids, when possible. In the worst case, we still use 6 symbols per atom id, but in case, for example, we see a three-symbol id, we know that the first symbol stands for the author and the other two for the symbol code. Assuming that atom ids are guaranteed to be bounded by delimiter symbols, we may agree to use `2=1+1`, `3=1+2`, `4=1+3`, `5=2+3` and `6=3+3` conventions. For example, suppose an atom id is encoded with five base64 characters: `0e5ZC`. Then, according to the `5=2+3` convention, the first two stand for the author code (up to $2^{2\times 6}$ values) and the next three stand for the symbol's number (up to $2^{3\times 6}$ values). Numerically, `0e` $= 0 \times 64^1 + 40 \times 64^0 + 48 = 88$ is the code for the author and `5ZC` $= 5 \times 64^2 + 35 \times 64^1 + 12 \times 64^0 + 48 = 22780$ is the serial number of the symbol among those contributed by that particular author to that particular page. (Unicode code point 88 corresponds to a Latin capital letter `X`. Code point 22780 corresponds to a hieroglyph 壼).

To express the aforementioned semantics of included/excluded bounds (see Sec. 3.2) and to guarantee reliable delimiters, we prepend every atom identifier with either `+` or `-`, denoting included or excluded bounding symbols respectively, e.g. `+0e5ZC`. The default value is `+`. We also allow atoms to be marked with mnemonic labels. Then, instead of a base64 representation of an atom id, we will use a single-quoted alphanumeric label, e.g. `+'SOME_VERSION'`.

While every author/yarn is encoded with an arbitrary Unicode char, it is convenient to make base64 representations somewhat meaningful semantically. Thus, instead of encoding "Alice" as `0e` (i.e. `88` or Unicode `X`), we will try to use such options as `Alc` (Unicode 갖), `Al` (Unicode ˣ) or simply `A` (Unicode `:`).

4.2 Text state/presentation specifier

Under the hood, the *ct* model has lots of version control related features and functions. Indeed, there are hundreds of ways to display an evolving text, and many of them are useful. The main bottleneck is the user's ability to perceive that data and to access that functionality. So, the core of our mission is to put that toolset within reach of an end user. Practically, we develop a small query language based on URI-embeddable expressions that will allow us to access the most of *ct*'s capacity right from the browser's address bar.

A *specifier* is a complete URI-embedded expression describing the desired state of the text and nuances of its decoration. A specifier contains a sequence of *parameters*. Each parameter affects a single aspect of text state or text presentation. A "state" parameter changes the actual text body delivered to the user, while a "presentation" parameter only adjusts its decoration (i.e. color, highlighting, strike-through, other marks). In this section we describe seven types of parameters: three state, three presentation and one mixed type.

The space/time unification helps us a lot. It lets all parameters follow the same syntax convention with minor variations. Every parameter starts with a special separator symbol (typically a sub- or general delimiter in terms of [8]). The separator defines the type of the parameter. The separator is followed by a sequence of zero or more atom identifiers and/or quoted labels.

Version is the first and the most basic state parameter. It defines the version of the text that is actually shown to the user. A version parameter employs exclamation mark `!` as a separator, normally followed by atom identifiers. Suppose, Alice wrote "Hallo wrld" and Bob corrected that to "Hello world". Thus Alice contributed 10 atoms (say, `Al01-Al0A`) and Bob contributed three (including one backspace, e.g. `Bo01-Bo03`). Then, the resulting version is `!Al0A+Bo03`.

Range is a state parameter specifying a fragment of a page that has to be delivered to the user. Range separator is `:` (a colon). In the example above, a range specifier `:Al01-Al06` initially points at "Hallo". Once Bob fixes the typo, the value of the same range changes to "Hello".

Branch is a parameter that allows to work with parallel versions of the same text. The separator is `=` (equal sign). A branch might be specified either with a label or with a yarn id, i.e. `='Branch'` may be interchangeable with `=Br`. The default "trunk" branch is addressed as `=`. The *ct* model allows to deal with branches in completely novel ways. In particular, it allows to merge (blend) branches in real time. Our syntax should let users access that functionality. In case multiple branches are specified, their contents are merged (blended), but all new edits go to the first mentioned branch. So, a specifier `='Draft'=` merges the trunk with the Draft branch, but all new edits go to the latter.

Fragment is a presentation parameter analogous to the range parameter. It specifies an area of interest within the delivered text. We re-use the standard URI fragment separator `#` (number sign). Important detail: the fragment part of URI is not reported to a HTTP server by a HTTP client (i.e. the browser). Hence, all corresponding actions are performed locally in the web browser (i.e. page scroll or range highlighting). In our example, `#Bo02-Al05` would show "Hello world" with "ell" selected or highlighted.

Baseline version is a presentation parameter pointing out which version is considered "the previous version". Thus, all changes that happened after that "previous" reference version should be highlighted. That is most useful when a user wants to see the changes that happened since his/her last visit, or otherwise compares two versions. This parameter employs `$` as a separator. Thus, to see a difference between two versions, Alice may use a specifier like `$Al06!Al0A`. That will deliver "Hallo wrld", with "wrld" highlighted.

Author parameter suggests to somehow mark/unmark contributions of certain authors. The separator is `@` (at sign). For example, `$Al08@-'Alice'` will unmark contributions made by Alice thus only leaving "e" and "o" highlighted, as those are contributed by Bob: "Hello world". Here we deviate from the general scheme of using full atom identifiers after a separator, as we only need to identify an author/yarn (the same as with branches). We may rewrite the same specifier as `$Al08@-Al`.

Change status is a mixed state/presentation parameter with a syntax somewhat deviating from the common pattern. It employs an asterisk `*` as a separator. Its mission is to filter/recover symbols based on their insertion/deletion status. For example, `$Al0A*+AlBo` shows all symbols inserted by Alice and removed by Bob since version `!Al0A`. Thus, the resulting text is "He~~a~~llo world", with "a" struck out, "e" and "o" highlighted.

Effectively, we created a small query language that controls state and presentation of a versioned text, points at ranges and versions, locates changes, navigates branches. As with any language, the expressive power comes from combining the primitives. We may easily imagine sophisticated but still comprehensible constructions, like:

```
http://server.dom/Proposal='Draft'$Alzu!Bo4Vk@-Bo#Alb8-AlyK
```

That means: "on a page named Proposal, within a branch named Draft, using version `Bo4Vk`, please highlight changes made since version `Alzu`, except for the changes made by Bob, and please select the range `Alb8–AlyK`". We do not expect every user to master this language. Composition of queries may be done by the GUI. Still, we see that this formal and compressed way of expressing versioning-related page state/presentation opens promising possibilities. One interesting example is the ability of specifiers to fully describe the current state of the edited page, including the current selection. If all changes of the state are reflected in the fragment part of the URI (rewriting fragment does not cause the page to reload), then the entire page state may be copied and sent by e-mail or IM to another person. An evolving text is almost like a river, in the sense that you cannot step into the same river twice. With specifiers, remote collaborators will have that possibility to be almost literally *on the same page*.

5 Scenarios

In this section we consider a hypothetic scenario of Happytown State University participating in a project that might be briefly described as a cross between OpenCourseWare and Wikipedia, collaboratively developed the git way. Collaborators from peer universities contribute academic information, including course materials, lecture notes and general articles. Users experience the system as a real-time wiki running in a browser. Each university hosts its own wiki.

That wiki also supports branches and distributed revision control to allow for parallel coexistence of drafts and working versions, on par with polished "canonical" public versions. Distributed revision control also allows to *federate* wikis of different universities. Users may pull changes from peer wikis in automated or manual fashion. Eventually, constant exchange of changes makes different wikis converge. Still, some pages may differ, because they are not merged yet or because of a conflict, e.g. in case Prof. Montague is unable to find common ground with Prof. Capuleti.

Voluntary import of changes allows to avoid Wikipedia-style edit warring and "the most persistent person wins" problem. It also acts as a soft variety of peer review, improving prestige of authors whose edits are widely accepted. While changes propagate from site to site, all authorship information is preserved intact. Academic prestige provides healthy incentives for participation, while direct spamming and self-promotion are countered by social filtering. Effectively, we consider a hypothetical bottom-up open-source academic publishing system.

5.1 Lecture and scribes

While assistant professor Alice delivers a lecture, appointed scribes transcribe it collaboratively in real-time by filling the lecture skeleton previously created by Alice. As the course is available on the web site both to peer universities and general public in real-time, scribes use a separate branch for their work: `http://ocw.happytown.edu/Lecture=Scribes`. Once scribes enter the URL, the branch is automatically created. They transcribe the lecture quite hastily. After the lecture, PhD student Bob polishes the text and merges it back into the university's trunk version. Thus, the public version is now updated and available to external audiences. Later, PhD student Fred of Fartown University decides to cite a passage from the lecture. He selects the passage and uses the link from his browser's address bar: `http://ocw.fartown.edu/Lecture#Bk-B7~`
Note that Fred uses the Fartown wiki which pulled content from Happytown.

5.2 Private remarks

Professor Carol oversees the lecture to see how well Alice is doing. Carol wants to see what scribes are recording to avoid getting out of sync with their version. Still, Carol wants to keep her remarks private. Thus, Carol blends `Scribes` with her own private branch by entering URL: `http://ocw.happytown.edu/Lecture =Notes=Scribes`. Now, the edits made by the scribes and her own changes are visible to Carol as a single merged text, updating in real time. Authorship is highlighted, one color per a yarn. Scribes cannot see the remarks made by the professor. Later, Carol will discuss the `Notes` with Alice and they will work on improving the text. A polished version of the branch will have to be merged back into the trunk. But, the edit history of the branch has lots of offhand remarks and back-and-forth editing which shouldn't go into the public history of the document. Thus, Carol *rebases* [4] the changes into the trunk, i.e. includes them by value, not by reference, leaving the edit history behind.

5.3 Back from vacation

PhD student Bob gets back from a conference and a vacation and logs into the system. He finds out that the project has advanced a lot since he departed. He loads the project status page. The changes made since his last visit are highlighted, contributions of different authors shown in different colors. At some point Bob decides to discuss the changes with post-doc Dave, who has authored some key new pieces of the text. Bob wants Dave to see exactly the same "blame map" highlighting as well. He picks the full URL of the current page state to paste it into an instant messaging client `http://ocw.happytown.edu/Lecture$Bo2j8` Then, he understands he should ask a focused question, so he selects a passage he is mostly concerned about. The URL changes to `http://ocw.happytown.edu/Lecture$Bo2j8#Bo64Q-D77j` now denoting the selection. Bob pastes the link into IM asking Dave for a clarification. This way he minimizes interruptions of context that would otherwise result from going and asking about "that change".

6 Conclusion

In this work, we bridged the *ct* model [14] of deep hypertext and the generic URI scheme [8]. We have shown that a simple and laconic convention may provide very fine-grained addressing for particular versions and/or segments of a changing text. Although the convention is *ct*-specific, it is rooted in the Lamport-Fidge time/event model and thus more predetermined than arbitrary.

We described several novel end-user scenarios for deep hypertext applications. Currently, distributed revision control is an expert-only domain. We have shown that such a functionality might be served to a regular academic user as well.

7 Acknowledgements

This work was partially supported by the EC FP7 project P2P-Next, grant #216217. Authors are grateful to David Hales for valuable feedback.

References

1. Deep hypertext: The Xanadu model. `http://www.xanadu.com/xuTheModel/`.
2. Edit warring on Wikipedia. `http://en.wikipedia.org/wiki/Wikipedia:Edit_warring`.
3. Google Wave protocol. `http://waveprotocol.org`.
4. Rebasing at "The Git Community Book". `http://book.git-scm.com/4_rebasing.html`.
5. The Unicode standard. version 6.0 - core specification. `http://www.unicode.org/versions/Unicode6.0.0/`.
6. Unix manual page for the patch utility. `man patch`.
7. XML Pointer Language (XPointer) Version 1.0. `http://www.w3.org/TR/WD-xptr`.
8. RFC 3986: Uniform Resource Identifier (URI): Generic Syntax, 2005.
9. Markov A.A. Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga. Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete, 15:135—156, 1906.
10. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: a distributed storage system for structured data. In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7, OSDI '06, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association.

11. C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. SIGMOD Rec., 18:399–407, 1989.
12. Colin Fidge. Logical time in distributed computing systems. Computer, 24(8):28–33, 1991.
13. Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. HTTP extensions for distributed authoring – WEBDAV. RFC 2518.
14. Victor Grishchenko. Deep hypertext with embedded revision control implemented in regular expressions. In Proceedings of the 6th International Symposium on Wikis and Open Collaboration, WikiSym '10, pages 3:1–3:10, New York, NY, USA, 2010. ACM.
15. Abdessamad Imine, Pascal Molli, Gérald Oster, and Michaël Rusinowitch. Proving correctness of transformation functions in real-time groupware. In ECSCW'03: Proceedings of the Eighth European Conference on Computer Supported Cooperative Work, pages 277–293, Norwell, MA, USA, 2003. Kluwer Academic Publishers.
16. E. E. Kim. An introduction to Purple. `http://eekim.com/software/purple/purple.html`.
17. E. Knutov, P. De Bra, and M. Pechenizkiy. Versioning in Adaptive Hypermedia. In Proceedings of the 1st DAH'2009 Workshop on Dynamic and Adaptive Hypertext: Generic Frameworks, Approaches and Techniques, pages 61–71, 2009.
18. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. Commun. ACM, 21(7):558–565, 1978.
19. Du Li and Rui Li. An admissibility-based operational transformation framework for collaborative editing systems. Comput. Supported Coop. Work, 19:1–43, February 2010.
20. Hermann Minkowski. Raum und Zeit. B. G. Teubner, Leipzig, 1909.
21. Theodor Holm Nelson. Literary Machines. Mindful Press, 1982.
22. Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang, and David Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. ACM Trans. Comput.-Hum. Interact., 5(1):63–108, 1998.
23. H. VandeSompel, M. Nelson, and R. Sanderson. HTTP framework for time-based access to resource states – Memento. `draft-vandesompel-memento-01`.

A set of adaptation patterns for expressing adaptive navigation in Adaptive Hypermedia

Nadjet Zemirline[1,2], Yolaine Bourda[1], and Chantal Reynaud[2]

[1] SUPELEC Systems Sciences (E3S) - Computer Science Department, France
(Nadjet.Zemirline, Yolaine.Bourda)@supelec.fr
[2] University of Paris-Sud XI, CNRS (LRI) & INRIA-Saclay/Projet Leo, France
Chantal.Reynaud@lri.fr

Abstract. This paper presents a set of 22 adaptation patterns, independent of any application domain and independent of any adaptation engine. They have been translated to LAG and GLAM adaptation languages in order to plug them on existing adaptation engines. Currently, they are used in the EAP framework, which allows defining complex adaptation strategies in Adaptive Hypermedia. We also propose a typology for the elementary adaptation patterns in order to facilitate their use and their understanding.

1 Introduction

Over the last decade, Adaptive Hypermedia (AH) have been under development [6], particularly in education [1], where learners get access to particular resources according to their knowledge, preferences and goals. Such access are proposed through the definition of adaptation, which is often considered as the most difficult part to author in AH [4, 2].

The definition of adaptation is made through expressing multiple adaptation strategies. An adaptation strategy specifies *which resources have to be proposed and how these resources will be proposed to a set of users who share the same characteristics* [9]. For example, users being theorist, textual and sequential, will have access only textual resources related to theory before those related to samples according to a depth-first navigational path on the relation successor.

Thereby, authors face numerous challenges when defining their adaptation strategies. The 1st challenge concerns the expression of adaptation strategies. The 2nd challenge concerns the reuse of adaptation strategies from one system to another one, and the expression of adaptation strategies independently of any AH System. To do so, the paradigm "*write once, use many*" [8] has been proposed. It endorses expressing adaptation at a high level, independently of any AHS, then translating this adaptation into a particular AHS. The 3rd challenge concerns the granularity in writing adaptation strategies. Its target is to avoid writing the common parts of adaptation strategies several times.

As shown in [9], till now, there have been no works concerning building complex adaptation strategies, independent of any AHS by combining simple adaptations.

So, in this paper, we present a set of 22 elementary adaptation patterns (EAP), easy to understand, independent of any application domain and also independent of any adaptation engine. They have been translated to LAG [2] and GLAM [7] in order to plug them to existing adaptation engines. Note that, LAG is already plug on multiple adaptation engines and GLAM proposes its adaptation engine. These EAP are used in the EAP framework [9]. The EAP framework enables authors to define complex adaptation strategies, at a high level and independent of any adaptation engine. It assists authors to instantiate our EAP on their domain model, thus to define elementary adaptations. Each elementary adaptation is associated to a user characteristic. As a user has multiple characteristics at a time, the framework proposes a semi-automatic combination process of elementary adaptations to compose complex adaptation strategies[3].

2 Description of an elementary adaptation pattern

We propose the following definition for elementary adaptation patterns (EAP), based on the definition of design patterns [5].

Definition 1. *An elementary adaptation pattern describes a generic solution for a generic elementary adaptation problem.*

The solution is independent from any language, and it exploits the characteristics of the domain model.

Definition 2. *A generic elementary adaptation problem describes a criterion to select resources to be proposed and a criterion to define in which order the selected resources are going to be proposed.*

In the following, we define fundamental criteria to select resources and to organize the selected resources on which EAP are based.

Criteria used to select resources. They are based on the structure of the domain model. We argue that the general description of a domain model includes the following elements: *a set of classes*, which must contain the class representing all the resources to be proposed, most often known as *Resource*, and the class representing all the domain concepts, most often known as *Concept*. *A set of relations between classes.* Each relation defines a graph on instances of classes on which it is defined. The graph have to be navigated in order to reach user goals. *A set of properties.*

Thereby, we have differentiated between criteria selecting resources and criteria defining a navigational path on relations. Our criteria for selecting resources are: (a) *their belonging to a class*, (b) *the values of some properties*, or (c) *the presence of a relation through the resources* or (d) *the presence of a relation through the concepts*. Furthermore, our criteria currently considered for defining a navigational path are either (a) *depth-first*, (b) *breadth-first*.

[3] The most difficult part of the combination is done automatically.

Criteria used to order the selected resources. In [9], we have studied over works defining adaptation methods, by giving a particular interest for adaptive navigation. We have retained four distinct and basic modes to select resources in a setting of adaptive navigation, as described below:

a - Selection only mode provides a set of resources based on a criterion. Only the selected resources are proposed to users, the other ones are not proposed. For example, we propose only textual resources.

b - Recommend selection mode provides multiple sets of resources (at least two) that include knowledge to specify which set should be recommended rather than the others. For example, to recommend definitions rather than examples. Both types of resources are accessible by users with distinct typographic indication to identify which resources are recommended.

c - Ordered selection mode provides multiple sets of resources (at least two), accompanied with knowledge to specify the order in which they must be presented. Only one set of resources is proposed at a time, and the resources of a particular set are not proposed until all the resources of all sets of higher priority have been viewed by the user. For example, concepts can be selected and ordered using the successor relation defined between concepts.

d - Alternate selection mode provides multiple sets of resources (at least two), accompanied with data that specifies the order in which they must be presented, knowing that only one set is presented to the user. For example, we propose textual resources when they are available, and audio resources in the absence of textual resources.

Table 1 presents the characteristics retained from [5] and used to describe EAP.

Name: is the name of the elementary adaptation pattern described. **Intent**: is a short statement about an elementary adaptation problem. It answers what is the elementary adaptation pattern supposed to do? i.e. what is its goal? Indeed, it indicates the way the resources are selected and the way they are presented. **Solution**: includes two elements: – *Expressions*: denote a set of resources to be proposed, and the conditions which have to be satisfied. These conditions can be represented in one or more logical expressions. Those to be considered simultaneously are gathered in the same expression, while excluded conditions are expressed in different expressions. – *Meta-expressions*: a binary relation between two expressions. Indeed, when using multiple expressions, we specify the way they have to be considered using meta-expressions. Let E_1, E_2 two expressions, we defined the following meta-expressions $E_1 \prec E_2$ to define an ordered selection mode. $E_1 \uplus E_2$ to define a recommended selection mode. $E_1 \mid E_2$ to define an alternate selection mode. **Constituents**: describe the elements of the domain model used.

Table 1: Description of elementary adaptation patterns

3 Organization of elementary adaptation patterns

Based on the criteria defined above, we have defined a library of 22 EAP. In order to be able to look easily over them, we have organized them in a tree where each leaf is an EAP (cf. Figure 1). The tree is read from left to right and each EAP is based simultaneously on:
1 - One of the 4 selection modes of resources to be proposed,
2 - One of the 3 elements of the domain model involved in the selection process
 - When the element is a relation, we consider also one of the 2 types of navigation through the resources or the concepts graph[4].

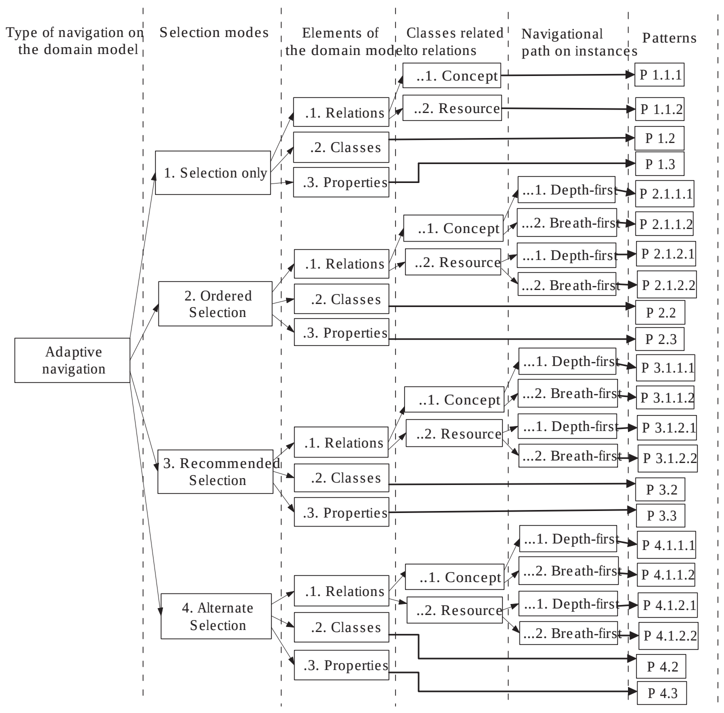


Fig. 1. Typology of elementary adaptation patterns

Just looking to the typology, we can deduce that the previous example of adaptation strategy (cf. Section 1) can be expressed using three separate EAP. Selecting select only textual resources is expressed using P1.3. Proposing ordered resources according to a successor relation is expressed using P2.1.2.1. Proposing resources related to theory before resources related to samples is expressed either

[4] The two navigation modes are applied for all the selection modes except for the *selection only* mode, which proposes a set of resources according to a criterion.

using P2.2 if theory and sample are modeled as two classes in the domain model, or using P2.3 if theory and sample are modeled as an attribute of the Resource class in the domain model.

We describe below our library of EAP per selection mode. Table 2 describes EAP using the selection only mode. P1.1.1, P1.1.2, P1.2 and P1.3 in Figure 1.

P 1.1.1	**Name**: Selection Only - Relation - Concept **Intent**: It proposes resources that are linked to concepts by *abstraction*, and where each concept can reach the concept named *goal* directly/indirectly using $relation_i$. **Solution**: Expression *E_1: linked-transitive(concept, goal, $relation_i$) $\wedge$ linked(r, concept, abstraction)* E_1 means that selected resources are linked to concepts using *abstraction*. The concepts can reach the *goal* using $relation_i$. **Constituents**: See row 1, row 2 in Table 6
P 1.1.2	**Name**: Selection Only - Relation - Resource **Intent**: It proposes resources that can reach the resource named *goal* directly or indirectly using $relation_i$. **Solution**: Expression *E_1: linked-transitive(r, goal, $relation_i$)* E_1 means that selected resources are linked to concepts using *abstraction* and they can reach the *goal* using $relation_i$. **Constituents**: See row 1, row 2 in Table 6
P 1.2	**Name**: Selection only - Classes **Intent**: It allows to select all resources of a specific type. **Solution**: Expression *E_1: instanceOf (r, $Class_1$)* E_1 means that selected resources are instances of the class $Class_1$. **Constituents**: See row 1, row 3 in Table 6
P 1.3	**Name**: Selection only- particular value of a property **Intent**: It allows to select resources according to some values of a property. **Solution**: Expression *E_1: characteristicOf(r, $property_i$, op, val)* E_1 means that selected resources must have the property $property_i$ and their value must satisfy the comparison test. **Constituents**: See row 1, row 4 in Table 6

Table 2: Elementary adaptation patterns using simple selection mode

Table 3 describes EAP using the ordered selection mode. P2.1.1.1, P2.1.1.2, P2.1.2.1, P2.1.2.2, P2.2 and P2.3 in Figure 1.

P 2.1.1.1	**Name**: Ordered Selection - Depth first- Relation - Concept **Intent**: It proposes resources according to a depth first navigational path on concepts. **Solution**:

Expression

- E_1*: linked(currentR, concept', abstraction)* $\wedge$ *linked-transitive(concept, goal, relation*$_i$*)* $\wedge$ *linked(r, concept, abstraction)* $\wedge$ *linked(concept, concept', relation*$_i$*)*
- E_2*: linked-transitive(concept, goal, relation*$_i$*)* $\wedge$ *linked(r, concept, abstraction)*

According to E$_1$ selected resources are linked to concepts using *abstraction*. The concepts can reach the *goal* using *relation$_i$* and are directly linked to current concept.
According to E$_2$ selected resources are linked to concepts using *abstraction*. The concepts can reach the *goal* using *relation$_i$*.
Meta-expressions
$E_1 \prec E_2$
According to this meta-expression, the set of resources selected by E$_1$ is proposed before the ones selected by E$_2$.
Constituents: See row 1, row 2 in Table 6

P 2.1.1.2

Name: Ordered Selection - Relation - Concept - breadth first
Intent: It proposes resources that are linked to concepts by *abstraction*, and where each concept can reach the concept named *goal* directly or indirectly using *relation$_i$* according to a depth first navigational path.
Solution:
Expression

- E_1*: linked-transitive(concept2, goal, relation*$_i$*)* $\wedge$ *linked(r, concept2, abstraction)* $\wedge$ *distance(concept2, origin, relation*$_i$*)* $\wedge$ *distance(concept, origin, relation*$_i$*)* $\wedge$ *linked(currentR, concept, abstraction)*
- E_2*: linked-transitive(concept, goal, relation*$_i$*)* $\wedge$ *linked(r, concept, abstraction)*

According to E$_1$ selected resources are linked to concepts using *abstraction*. The concepts are linked to the *goal* using *relation$_i$* and have the same distance of the concept which is an abstraction of current resource from the first selected resource.
According to E$_2$ selected resources are linked to concepts using *abstraction*. The concepts can reach the *goal* using *relation$_i$*.
Meta-expressions
$E_1 \prec E_2$
According to this meta-expression, the set of resources selected by E$_1$ is proposed before the ones selected by E$_2$.
Constituents: See row 1, row 2 in Table 6

P 2.1.2.1

Name: Ordered Selection - Relation - Resource - Depth-first
Intent: It proposes resources that can reach the resource named *goal* directly or indirectly using *relation$_i$* according to a depth first navigational path.
Solution:
Expression

- E_1*: linked-transitive(resource, goal, relation*$_i$*)* $\wedge$ *linked(currentR, resource, relation*$_i$*)*
- E_2*: linked-transitive(r, goal, relation*$_i$*)*

Meta-expressions
$E_1 \prec E_2$

	According to this meta-expression, the set of resources selected by E_1 is proposed before the ones selected by E_2. **Constituents**: See row 1, row 2 in Table 6
P 2.1.2.2	**Name**: Ordered Selection - Relation - Resource - Breadth-first **Intent**: It proposes resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a breadth first navigational path. **Solution**: Expression – E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *distance(resource, origin,* $relation_i$*)* $\wedge$ *distance(currentR, origin,* $relation_i$*)* – E_2: *linked-transitive(r, goal,* $relation_i$*)* Meta-expressions $E_1 \prec E_2$ According to this meta-expression, the set of resources selected by E_1 is proposed before the ones selected by E_2. **Constituents**: See row 1, row 2 in Table 6
P 2.2	**Name**: Ordered Selection - Classes **Intent**: It proposes ordered resources belonging only to subclasses of the class Resource. **Solution**: Expression - E_1: *instanceOf (r,* $Class_1$*)* - .... - E_n: *instanceOf (r,* $Class_n$*)* Meta-expressions $E_i \prec E_j$, i < j, i = 1..n and j = 1..n. According to this meta-expression, the set of resources selected by E_i is proposed before the ones selected by E_j (i < j). **Constituents**: See row 1, row 3 in Table 6
P 2.3	**Name**: Ordered Selection - Properties **Intent**: It proposes ordered resources that satisfy some values of the property $property_i$. **Solution**: Expression - E_1: *characteristicOf(r,* $property_i$ *, op,* val_1*)* - .... - E_n: *characteristicOf(r,* $property_i$ *, op,* val_n*)* Meta-expressions $E_i \prec E_j$, i < j, i = 1..n and j = 1..n. According to this meta-expression, the set of resources selected by E_i is proposed before the ones selected by E_j (i < j). **Constituents**: See row 1, row 4 in Table 6

Table 3: Elementary adaptation patterns using ordered selection mode

Table 4 describes EAP using the recommended selection mode. P3.1.1.1, P3.1.1.2, P3.1.2.1, P3.1.2.2, P3.2 and P3.3 in Figure 1.

P 3.1.1.1	**Name**: Recommended Selection - Relation - Concept- Depth first **Intent**: It proposes recommended resources that are linked to concepts by *abstraction*, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$ according to a depth-first navigational. **Solution**: Expression – E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2, abstraction)* $\wedge$ *linked(concept, concept2,* $relation_i$*)* $\wedge$ *linked(currentResource, concept, abstraction)* – E_2: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept, abstraction)* Meta-expressions $E_1 \uplus E_2$ According to this meta-expression, the set of resources selected by E_1 is recommended rather than the ones selected by E_2. **Constituents**: See row 1, row 2 in Table 6
P 3.1.1.2	**Name**: Recommended Selection - Relation - Concept - breadth first **Intent**: It proposes recommended resources that are linked to concepts by *abstraction*, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path. **Solution**: Expression – E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2, abstraction)* $\wedge$ *distance(concept2, origin,* $relation_i$*)* $\wedge$ *distance(concept, origin,* $relation_i$*)* $\wedge$ *linked(currentResource, concept, abstraction)* – E_2: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept, abstraction)* Meta-expressions $E_3 \uplus E_2$ According to this meta-expression, the set of resources selected by E_1 is recommended rather than the ones selected by E_2. **Constituents**: See row 1, row 2 in Table 6
P 3.1.2.1	**Name**: Recommended Selection - Relation - Resource - Depth-first **Intent**: It proposes recommended resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path. **Solution**: Expression – E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *linked(currentResource, resource,* $relation_i$*)* – E_2: *linked-transitive(r, goal,* $relation_i$*)* Meta-expressions $E_4 \uplus E_5$ According to this meta-expression, the set of resources selected by E_1 is recommended rather than the ones selected by E_2. **Constituents**: See row 1, row 2 in Table 6
P 3.1.2.2	**Name**: Recommended Selection - Relation - Resource - Breadth-first

	Intent: It proposes recommended resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a breadth first navigational path. **Solution**: Expression – E_1: *linked-transitive(resource, goal, $relation_i$) $\wedge$ distance(resource, origin, $relation_i$) $\wedge$ distance(currentResource, origin, $relation_i$)* – E_2: *linked-transitive(r, goal, $relation_i$)* Meta-expressions $E_1 \uplus E_2$ According to this meta-expression, the set of resources selected by E$_1$ is recommended rather than the ones selected by E$_2$. **Constituents**: See row 1, row 2 in Table 6
P 3.2	**Name**: Recommended Selection - Classes **Intent**: It proposes recommended resources according to their type. **Solution**: Expression – E_1: *instanceOf (r, $Class_1$)* – ... – E_n: *instanceOf (r, $Class_n$)* Meta-expressions $E_i \uplus E_j$, i < j, i = 1..n and j = 1..n. According to this meta-expression, the set of resources selected by E$_i$ is recommended rather than the ones selected by E$_j$ (i < j). **Constituents**: See row 1, row 3 in Table 6
P 3.3	**Name**: Recommended Selection - Properties **Intent**: It proposes resources that satisfy some values of the property $property_i$. **Solution**: Expression – E_1: *characteristicOf(r, $property_i$, op, val_1)* – .... – E_n: *characteristicOf(r, $property_i$, op, val_n)* Meta-expressions $E_i \uplus E_j$, i < j, i = 1..n and j = 1..n. According to this meta-expression, the set of resources selected by E$_i$ is recommended rather than the ones selected by E$_j$ (i < j). **Constituents**: See row 1, row 4 in Table 6

Table 4: Elementary adaptation patterns using recommended selection

Table 5 describes EAP using the alternate selection mode. P4.1.1.1, P4.1.1.2, P4.1.2.1, P4.1.2.2, P4.2 and P4.3 in Figure 1.

P 4.1.1.1	**Name**: Alternate Selection - Relation - Concept- Depth first

Intent: It proposes alternate resources that are linked to concepts by *abstraction*, wher each concept can reach the concept named *goal* directly/indirectly using $relation_i$ according to a depth-first navigational.

Solution:

Expression

- E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2, abstraction)* $\wedge$ *linked(concept, concept2,* $relation_i$*)* $\wedge$ *linked(currentResource, concept, abstraction)*
- E_2: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept, abstraction)*

Meta-expressions

$E_1 \mid E_2$

According to this meta-expression, the set of resources selected by E_1 is alternate of the ones selected by E_2.

Constituents: See row 1, row 2 in Table 6

Pattern 4.1.1.2

Name: Alternate Selection - Relation - Concept - breadth first

Intent: It proposes alternate resources that are linked to concepts by *abstraction*, and where each concept can reach the concept named *goal* directly/indirectly using $relation_i$ according to a depth first navigational path.

Solution:

Expression

- E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2, abstraction)* $\wedge$ *distance(concept2, origin,* $relation_i$*)* $\wedge$ *distance(concept, origin,* $relation_i$*)* $\wedge$ *linked(currentResource, concept, abstraction)*
- E_2: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept, abstraction)*

Meta-expressions

$E_1 \mid E_2$

According to this meta-expression, the set of resources selected by E_1 is alternate of the ones selected by E_2.

Constituents: See row 1, row 2 in Table 6

Pattern 4.1.2.1

Name: Alternate Selection - Relation - Resource - Depth-first

Intent: It proposes alternate resources that can reach the resource named *goal* directly/indirectly using $relation_i$ according to a depth first navigational path.

Solution:

Expression

- E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *linked(currentResource, resource,* $relation_i$*)*
- E_2: *linked-transitive(r, goal,* $relation_i$*)*

Meta-expressions

$E_1 \mid E_2$

According to this meta-expression, the set of resources selected by E_1 is alternate of the ones selected by E_2.

Constituents: See row 1, row 2 in Table 6

Pattern 4.1.2.2

Name: Alternate Selection - Relation - Resource - Breadth-first

Intent: It proposes alternate resources that can reach the resource named *goal* directly/indirectly using $relation_i$ according to a breadth first navigational path.

	Solution: Expression – E_1: *linked-transitive(resource, goal, relation$_i$) $\wedge$ distance(resource, origin, relation$_i$) $\wedge$ distance(currentResource, origin, relation$_i$)* – E_2: *linked-transitive(r, goal, relation$_i$)* Meta-expressions $E_1 \mid E_2$ According to this meta-expression, the set of resources selected by E_1 is alternate of the ones selected by E_2. **Constituents**: See row 1, row 2 in Table 6
Pattern 4.2	**Name**: Alternate Selection - Classes **Intent**: It proposes alternative resources according to their type. **Solution**: Expression – E_1: *instanceOf (r, Class$_1$)* – ... – E_n: *instanceOf (r, Class$_n$)* Meta-expressions $E_i \mid E_j$, i < j, i = 1..n and j = 1..n. According to this meta-expression, the set of resources selected by E_i is an alternative of the ones selected by E_j (i < j). **Constituents**: See row 1, row 3 in Table 6
Pattern 4.3	**Name**: Alternate Selection - Properties **Intent**: It proposes alternative resources, where each of them satisfy a value of the property *property$_i$*. **Solution**: Expression – E_1: *characteristicOf(r, property$_i$, op, val$_1$)* – .... – E_n: *characteristicOf(r, property$_i$, op, val$_n$)* Meta-expressions $E_i \mid E_j$, i < j, i = 1..n and j = 1..n. According to this meta-expression, the set of resources selected by E_i is alternate of the ones selected by E_j (i < j). **Constituents**: See row 1, row 4 in Table 6

Table 5: Elementary adaptation patterns using alternate selection mode

row1	r: represents an instance of the class *Resource* or of one of its specializations.
row2	concept: represents an instance of the *Concept* class. goal: represents the goal to reach. It is an instance of *Concept* class. currentResource: represents an instance of the current resource. relation$_i$: represents a relation between instances of the *Concept* class. origin: represents the first resources proposed to the user.

	abstraction: represents a relation between an instance of the *Concept* class and instance(s) of the *Resource* class or of one of its specializations.
row3	Class$_i$: represents a subclass of the *Resource* class.
row4	property$_i$: represents a property of the *Resource* class. val: represents a possible value for the property *property$_i$*.

Table 6: Constituents used in elementary adaptation patterns

4 Discussion and conclusion

Concerning the domain model, we argue that whatever the domain model, it is composed of a set of classes, of properties and relations [7, 4, 2]. Therefore our EAP are independent of any domain model.

Concerning the adaptive navigation, we have conducted a study [9] of supported types of adaptive navigation and we have included all of them in our EAP. In case, we didn't consider a type of adaptive navigation, our EAP are extensible to support missing types of adaptive navigation.

We also argue that a user model is always composed of a set of characteristics. For any user model, our framework is generic to consider any user characteristics. The EAP framework enables to associate a value of a user characteristic to an instantiation of EAP. Afterward, it generate complex adaptation strategies basing on combinations of user characteristics.

References

1. P. Brusilovsky, J. Eklund, and E. Schwarz, "Web-based education for all: a tool for development adaptive courseware," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 291–300, April 1998.
2. A. Cristea and L. Calvi, "The three layers of adaptation granularity," in *Proceedings of the 9th int. conference on User modeling*, ser. UM'03. Berlin, Heidelberg: Springer, 2003, pp. 4–14.
3. P. Brusilovsky, "Adaptive navigation support for open corpus hypermedia systems," in *AH*, 2008, pp. 6–8.
4. P. De Bra, D. Smits, and N. Stash, "Creating and delivering adaptive courses with aha!" in *1st Eur. Conf. on Technology Enhanced Learning*, ser. 0302-9743, LNCS, Ed. Springer, 2006, pp. 21–33.
5. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
6. E. Knutov, P. De Bra, and M. Pechenizkiy, "Ah 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques," *New Rev. Hypermedia Multimedia*, vol. 15, pp. 5–38, April 2009.
7. C. Jacquiot, Y. Bourda, F. Popineau, A. Delteil, and C. Reynaud, "Glam: A generic layered adaptation model for adaptive hypermedia systems," in *AH*, 2006, pp. 131–140.
8. C. Stewart, A. Cristea, T. Brailsford, and H. Ashman, "Authoring once, delivering many: Creating reusable adaptive courseware," in *WBE 2005 Conference*, 2005.
9. N. Zemirline, Y. Bourda, and C. Reynaud, "Expressing adaptation strategies using adaptation patterns," in *TLT. Accepted, to appear*, 2011.

Adaptive Hypermedia Systems Analysis Approach by Means of the GAF Framework

Evgeny Knutov, Paul De Bra, and Mykola Pechenizkiy

Department of Computer Science, Eindhoven University of Technology,
P.O. Box 513, 5600 MB, Eindhoven, the Netherlands
e.knutov@tue.nl, debra@win.tue.nl, m.pechenizkiy@tue.nl

Abstract. Adaptive Hypermedia Systems (AHS) have long been concentrating on adaptive guidance of links between domain concepts with lots of custom developments and ad-hoc implementations. Here we consider a formalization approach to AHS composition and design by defining building blocks' interfaces and presenting corresponding dependencies by means of the GAF framework. This helps to identify system design guidelines and start building adaptive system from scratch as well as analyze adaptive system behaviour, architecture and risks involved.

1 Introduction

Since the most cited Adaptive Hypermedia (AH) model AHAM [1] new terms, definitions and models have been introduced and realized in prototypes. Most AH models focus on a layered architecture and concentrate on adaptation to the linking and navigation between concepts of a domain. With the exploding popularity of the Web searching rather than linking, or Recommender systems (RS) to rank relevant content and provide personalized information the area of AHS has gained a lot. The Generic Adaptation Framework (GAF)[1] research project aims to develop a new reference model for the adaptive hypermedia research field. The new model considers new developments, techniques and methodologies in the areas of adaptive hypermedia and adjacent fields. Besides GAF concerns the detailed system analysis in terms of AHS building blocks, connections and dependencies, approaches that can be used to implement such a system.

GAF conceptual scheme of the layered structure is presented in Figure 1. It aligns the order of the layers in the system according to the classification of AH methods and techniques [5]. Though this order represents the basic understanding of the adaptation questions, every particular system may vary or even omit some of these, thus leading to a different composition of the system layers determined by the different adaptation idea behind this (adaptive eLearning application, Recommender System, etc.). We believe that in order to couple, align, sort and arrange the layers of such a system (both the generic model or some particular domain focused implementation) one should keep in mind an adaptation process scenario (partially considered as use-cases in [4]) that will partially determine the layer arrangement and to some extent will define the mandatory and optional elements and drive the system design.

[1] `http://www.win.tue.nl/~eknutov/gaf.html`

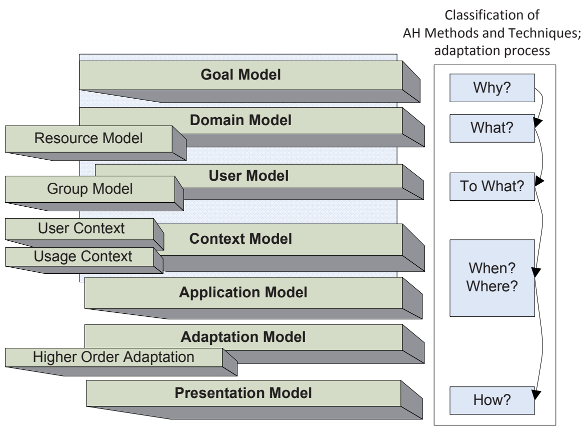


Fig. 1. Conceptual scheme of GAF layered structure

2 AHS Analysis Approach

As thoroughly investigated in [7] the evaluation of AH systems plays an important role. The described layered evaluation provides the description of the system functionality and helps to solve many related problems. In our work we consider a more formalized and specific system analysis approach by taking up systems' block composition scenarios, interfaces. Thus we define dependencies between models, methods they use to communicate with each other and particular implementations (based on usage scenarios). As a reference we took the approach from [3]. The main steps of such an analysis are presented in Figure 2. By scenarios here we mean framework use-cases (adaptive search, adaptive eLearning, recommender system, etc.), mostly covered in [4]. These scenarios are represented by 'sequence charts' and are constructed using GAF layers. We also consider system specific aspects and AHS building blocks composition which impacts the system architecture, such as event-driven system or service oriented or these two together.

As a result of this approach we would have elementary base concerns of AHS, which would explain mandatory and optional building blocks of the system, trade-off available, mostly concerning optional elements of AHS, and the dependencies involved presented as table. We will elaborate the approach further and explain it through the example of the Domain Model (DM).

3 AHS Models Analysis Approach: DM example

Hereafter we elaborate the analysis approach and consider the AHS DM. In Figure 3 we show an example of DM interface dependencies. Analyzing it down further we comprise the dependency table of building blocks' interfaces (such as Domain, Use, Resource, Context models), scenarios of how these models are used and which type

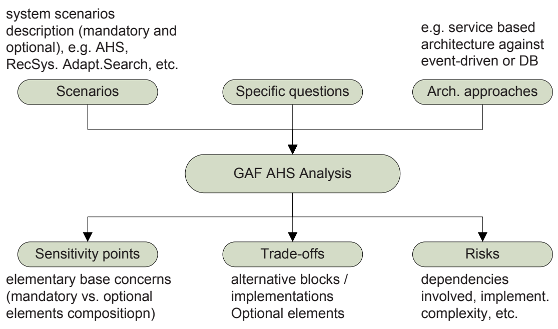

system scenarios description (mandatory and optional), e.g. AHS, RecSys. Adapt.Search, etc.
e.g. service based architecture against event-driven or DB
Scenarios
Specific questions
Arch. approaches
GAF AHS Analysis
Sensitivity points
Trade-offs
Risks
elementary base concerns (mandatory vs. optional elements compositiopn)
alternative blocks / implementations Optional elements
dependencies involved, implement. complexity, etc.

Fig. 2. AHS analysis approach

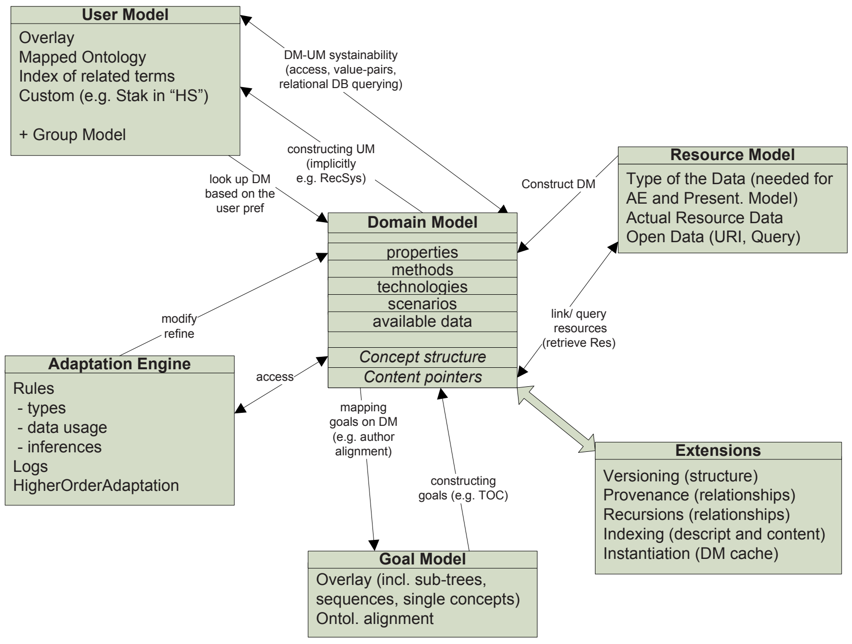

User Model
Overlay
Mapped Ontology
Index of related terms
Custom (e.g. Stak in "HS")
+ Group Model
DM-UM systainability (access, value-pairs, relational DB querying)
constructing UM (implicitly e.g. RecSys)
look up DM based on the user pref
Construct DM
Resource Model
Type of the Data (needed for AE and Present. Model)
Actual Resource Data
Open Data (URI, Query)
Domain Model
properties
methods
technologies
scenarios
available data
Concept structure
Content pointers
modify refine
link/ query resources (retrieve Res)
Adaptation Engine
Rules
- types
- data usage
- inferences
Logs
HigherOrderAdaptation
access
mapping goals on DM (e.g. author alignment)
constructing goals (e.g. TOC)
Extensions
Versioning (structure)
Provenance (relationships)
Recursions (relationships)
Indexing (descript and content)
Instantiation (DM cache)
Goal Model
Overlay (incl. sub-trees, sequences, single concepts)
Ontol. alignment

Fig. 3. Domain Model interface dependencies

of system is being described (AHS, Adaptive eLearning, Recommender System, etc.), possible technologies to implement it (Data Bases, OWL ontologies for semantic web enabled systems, TF-IDF index for search, etc.). As a result we'll have a detailed picture of the system components, evaluated against the reference model (GAF), which will help to identify all pros and cons.

Considering any arbitrary DM properties and interfaces we analyze them against the following properties and methods of the reference structure (see Figure 4 for de-

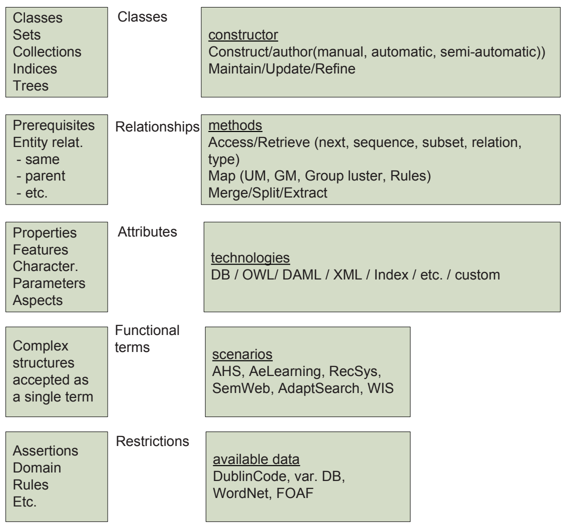

Classes
Sets
Collections
Indices
Trees
Classes
constructor
Construct/author(manual, automatic, semi-automatic))
Maintain/Update/Refine
Prerequisites
Entity relat.
- same
- parent
- etc.
Relationships
methods
Access/Retrieve (next, sequence, subset, relation, type)
Map (UM, GM, Group luster, Rules)
Merge/Split/Extract
Properties
Features
Character.
Parameters
Aspects
Attributes
technologies
DB / OWL/ DAML / XML / Index / etc. / custom
Complex structures accepted as a single term
Functional terms
scenarios
AHS, AeLearning, RecSys, SemWeb, AdaptSearch, WIS
Assertions
Domain
Rules
Etc.
Restrictions
available data
DublinCode, var. DB, WordNet, FOAF

Fig. 4. Domain Model abstraction class

Table 1. partial GAF blocks high-level dependencies: DM example

DM properties and methods	Scenario	Resource Model	Adaptation Engine	User Modelling
concept tree	conventional AHS eLearning	content pages/frames	ECA reasoning, prerequisites relations	UM overlay
feature space	recommender system	datasets	promotions and ranking mechanisms	implicit user profiling
index	adaptive search	WWW	ranking	implicit user profiling

tails). The major division here concerns methods and properties of the abstract Domain Model class. Further we distinguish classes (like sets or collections of concepts or concept maps, indices, trees, etc.), relationships (which are conventionally constituted by the ontology relationships), attributes of the concepts (e.g feature space, properties, characteristics, etc.), then functional terms which are denoted by complex structures usually treated as a single term, and restrictions defined by assertions or some specific domain rules.

Methods can be defined by constructors used to author DM as well as refine, maintain or update it. Major DM methods describe the access and retrieve procedures mainly called by User Model (UM), Resource model (RM) and Adaptation Engine (AE) to access the conceptual structure and query corresponding content. We also define mapping methods which are used to maintain structure sustainability especially in overlay type of models or ontology mapping for instance. These mappings (or alignments) can be done between DM and User, Goals, Groups models and Rules sets. Additionally we have methods to merge, split and extract sub-models of DM, which can be used in distributed domain modelling or open corpus adaptation.

DM scenarios describe the system behaviour in terms of functional flow and user interaction. We have described most prominent use-cases of such a framework compliance with different types of systems in [4]. Thus the DM usage in different cases could be analyzed against these reference scenarios.

Finally we have a number of particular technologies to work with DM and associated or cross-technology data available to start modelling (e.g Dublin Core to devise adaptive eLearning application or a dataset feature list to devise recommender system or adaptive search portal). This may remind us of the UML notion used in [6] to formalize the AHS modelling, however we define more strict dependencies in the GAF formalization through defining interfaces, methods and scenarios, besides we use it to analyze system, identify alternatives and be able to compare and assess other systems in terms of the GAF framework. Table 1 presents high-level dependencies between DM properties and methods, scenarios and other AHS' models. This is just to give an idea of our approach, ideally these dependencies would be described in meticulous details, parametrizing abstract DM interfaces and to some extent show concrete technology or implementation approach for each of these models' interfaces.

4 Summarizing Implications of the Analysis Approach

Here we would like to summarize the major implications of our approach and anticipated benefits.

– Reference structures — being a reference model GAF and detailed dependencies of its layers will serve as an ideal starting point for AH system designers and researches in the field.
– Complexity and Performance — defining a number of dependencies and known technologies would give an impression of the system complexity.
– Compatibility and Compliance — compliance description ([4]) provides the description of use-cases and application scenarios of the GAF framework.
– Modifiability — trade-off between blocks or modules' alternatives will show the modification possibilities, or further system extensions.

5 Conclusions and Future Work

The coming years will bring more use-cases of how AHS can provide adaptation and personalization, what techniques will be introduced, and what research areas will introduce new technologies in its evolution. So far a study of existing adaptation and personalization approaches was done to comply with the layered structure of adaptive information systems, which raised the problem of system composition and design analysis. We try to solve this problem using a classical software architecture analysis approach extending it with adaptation framework specific questions and interface dependencies in order to meticulously analyze any adaptive system in terms of the GAF framework.

At the same time evaluating the proposed general-purpose AHS architecture (GAF framework) against recommender systems [2] has shown that the GAF architecture is sufficiently generic to accommodate the description of different personalization approaches including recommenders, as well as provide the flexibility of both AH and RS in one go by building a custom system with the GAF building blocks. The real though not very meticulous case study has proven our points. It has given us new challenges to investigate the applicability of new approaches, as well as new developments in adaptive information systems which will allow to decide on the system composition at the implementation level and this is where one would need the AHS analysis.

6 Acknowledgements

This work has been supported by the *NWO GAF: Generic Adaptation Framework* project.

References

1. P. De Bra, G.-J. Houben, and H. Wu. AHAM: a Dexter-Based Reference Model for Adaptive Hypermedia. In *Hypertext*, pages 147–156. ACM, 1999.
2. J. Hannon, E. Knutov, P. D. Bra, M. Pechenizkiy, K. McCarthy, and B. Smyth. Bridging Recommendation and Adaptation: Generic Adaptation Framework - Twittomender compliance case-study. In *to appaear in Proc. of the 2nd International Workshop on Dynamic and Adaptive Hypertext (DAH'11)*, 2011.
3. R. Kazman, M. Klein, and P. Clements. ATAM: Method for architecture evaluation. Technical Report ESC-TR-2000-004, Carnegie Mellon, Software Engineering Institute, 2000.
4. E. Knutov, P. D. Bra, and M. Pechenizkiy. Generic adaptation framework: a process-oriented perspective. *J. Digit. Inf.*, 12(1), 2011.
5. E. Knutov, P. De Bra, and M. Pechenizkiy. AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. *New Rev. Hypermedia Multimedia*, 15(1):5–38, 2009.
6. N. Koch. *"Software engineering for adaptive hypermedia systems"*. PhD thesis, Ludwig-Maximilians University of Munich, Munich, Germany, 2001.
7. A. Paramythis, S. Weibelzahl, and J. Masthoff. Layered evaluation of interactive adaptive systems: framework and formative methods. *User Modeling and User-Adapted Interaction*, 20:383–453, December 2010.

Open Corpus Adaptation++ in GALE: Friend or Foe?

David Smits and Paul De Bra

Faculty of Mathematics and Computer Science, Eindhoven University of Technology,
Postbus 513, 5600 MB Eindhoven, The Netherlands
`d.smits@tue.nl, debra@win.tue.nl`

Abstract. "Open" has quickly become the hottest topic in any field related to *information*, including open government data, open learning resources, open user models, … Open Corpus Adaptation has been defined as the ability to perform adaptation to resources located anywhere on the Web. This leaves the definition of and control over the adaptation in a central place. GALE adds the ability to have the adaptation (definition) distributed over the Web. In this paper we describe how GALE achieves this functionality and we raise the question whether this is actually a desired feature or potentially a dangerous addition with unintended consequences.

1 Introduction and Motivation

Using hypertext to open up and link all available information was first suggested by Ted Nelson when introducing Xanadu (see http://www.xanadu.net/) and became a reality soon after the introduction of the Web. The initial Web was a "safe" environment, where all information was static. The browser could download any web page and display it, and the user would be assured that this would not have any side effects. Since then the (on-line) world has become much more dynamic. Our data resides "in the cloud", processing is done "in the cloud", but even when just accessing websites they know (remember) who we are, and they may cause our browser to execute code we have no control over.

So far *adaptive* hypermedia applications have been "safe": an adaptive application is served by a single *adaptive hypermedia system* (AHS), providing adaptation to local resources, and storing user-related information in a local database. These applications are also "closed". Initiatives to open up AHS have so far approached two aspects: 1) the *user model* has become distributed [1], integrating information coming from many different adaptive and non-adaptive applications, including social networks, and 2) the *resources* have become distributed in *open corpus adaptive hypermedia* [2]. Distributing user model and resources has also been a goal in the GRAPPLE project (http://www.grapple-project.org/) in which GALE (the GRAPPLE Adaptive Learning Environment) was developed. Within GRAPPLE it was foreseen that the definition of the adaptation would be kept centralized, created through a graphical authoring toolset GAT (GRAPPLE Authoring Tool) [3]. GALE has been designed to be able to perform to resources that can be loaded from anywhere on the Web (retrieved through HTTP). This leads to a seemingly strange situation: an author defines adaptation for a resource in GAT and GALE performs adaptation to that resource that is created by someone else, located anywhere in the world, without the author of the resource having any influence on the adaptation that will be performed to that resource. Would it not be logical to enable authors of resources to also define the adaptation (and user model updates) associated with that resource? This is exactly what the *Open Corpus Service* in GALE allows. In Sect. 2 we describe this "open corpus adaptation++" and then (Sect. 3) we discuss the feasibility of actually uptake of this functionality and the potential dangers involved.

2 Open Corpus Adaptation++ in GALE

Figure 1 below shows the overall architecture of GALE. This architecture is "distributed" around an internal Event Bus.

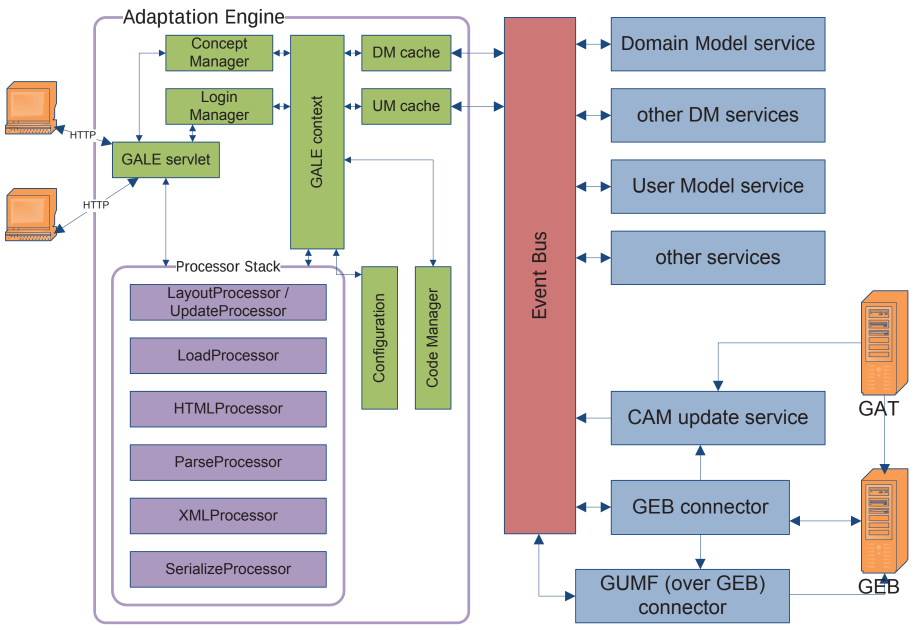


Fig.1. Core GALE architecture

In "standard" use an application is defined by an author and added to GALE through the "CAM update service", resulting in a *domain model* (DM) which in GALE contains the conceptual structure and the adaptation for the application. The GALE event bus can connect different DM services with the common adaptation engine. It can also connect different *user model* services, including an internal GALE user model service and an external GRAPPLE User Model Framework GUMF. In this paper we concentrate on the Open Corpus Service, which is a DM service.

A domain model in GALE is defined using the GAM language (GALE application model). The authoring process normally results in a set of *concepts* with for each concept the associated GAM code that defines *properties*, *user model attributes* and *event code* for the concept and its attributes. All requests to GALE normally specify a concept (not a web page). When the concept specification refers to a concept on an external server (the concept is requested from another server through HTTP) the Open Corpus service retrieves that concept and scans the file for a <meta> element with a 'name' attribute with value 'gale.dm'. When no information for the current concept is found, the Open Corpus service searches for files called concept.gdom and concept.gam (where concept stands for the actual concept name). It does so from the current path in the URL up to the root of the server specified. The first description found on the current concept is used.

Below is an example http://gale.win.tue.nl/elearning.xhtml (taken from [4]) with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns=http://www.w3.org/1999/xhtml
      xmlns:gale="http://gale.tue.nl/adaptation">
    <head>
      <meta name="gale.dm" content="
      { #[visited]:Integer `0` {
        event `if (${#suitability} && ${#read} < 100)
                     #{#read, 100};
               else if (!${#suitability} && ${#read} < 35)
                     #{#read, 35};`}
        #knowledge:Integer !`GaleUtil.avg(new Object[]
        {${<=(parent)#knowledge},${#read}}).intValue()`
        #[read]:Integer `0`
        #suitability:Boolean `true`
        event `#{#visited, ${#visited}+1};` } " />
    </head>
    <body>
    <p>This page is a placeholder for the elearning
       concept.</p>
    </body>
</html>
```

We don't describe the details of the GAM syntax and semantics here, but only briefly explain the example code:

- The code `event `#{#visited, ${#visited}+1};` }` means that when the concept is accessed the value of the "visited" attribute in increased by 1.
- The attribute "visited" is an integer, and when its value changes its event code is execute which updates the "read" attribute.
- The attribute "read" is also an integer.
- The attribute "knowledge" is an integer which is not stored but calculated from the "read" value and the "knowledge" value of the children of the "elearning" concept.
- The attribute "suitability" is a Boolean, which is "true" by default. This too is not stored but calculated when needed. If there were prerequisites for the "elearning" concept there would be an expression that defines the condition for the concept to become suitable.

Another “page” can “inherit” this adaptation (GAM) code as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns=http://www.w3.org/1999/xhtml
      xmlns:gale="http://gale.tue.nl/adaptation">
    <head>
        <meta name="gale.dm" content= {->(extends)
           http://gale.win.tue.nl/elearning.xhtml}" />
    </head>
    <body>
    <p>This page uses the elearning template.</p>
    </body>
</html>
```

When a whole application domain is stored in a single file the “meta” element for the concepts/pages would look like:

```
<meta name='gale.dm' content='redirect:course.gam' />
```

and the file “course.gam” might have contents like:

```
welcome.xhtml {
    ->(extends)http://gale.win.tue.nl/elearning.xhtml
    ->(extends)layout.xhtml
    <-(parent)gale.xhtml
    <-(parent)gat.xhtml
}
gale.xhtml {
    ->(extends)welcome.xhtml
  ->(parent)welcome.xhtml
}
gat.xhtml {
    -> (extends)welcome.xhtml
    ->(parent)welcome.xhtml
}
layout.xhtml {
    #layout:String `
    <struct cols="250px;*">
      <view name="static-tree-view" />
      <struct rows="60px;*;40px">
        <view name="file-view" file="gale:/header.xhtml" />
        <content />
        <p><hr />Next suggested concept to study:
             <view name="next-view" /></p>
      </struct>
  </struct> `
}
```

Again we do not explain this code but just illustrate that code can be shared between different concepts/pages, and can be placed in individual files or combined into a single GAM file.

When GALE retrieves “open corpus GAM definitions” it treats them just like a locally stored definition: the concepts are created, user model information is stored and updated, and the adaptation of other concepts (and the retrieved concepts themselves) can depend on user model values for both these external and internal concepts. The event code in GAM is essentially arbitrary Java code. This has potentially serious implications which we discuss in the next section.

3 The implications of Open Corpus Adaptation++

When "dynamic" content was first introduced on the Web it came with significant security concerns. To illustrate:

- Browser plug-ins consist of executable code that can potentially harm the end-user's computer. It has full access to all resources to which the browser has access. A harmful plug-in can not only crash the browser but also wipe the user's hard drive, send spam messages, search for critical personal data on the hard drive like credit card numbers and transfer that to a criminal organization, etc.
- Scripting code can be made somewhat less dangerous depending on what the scripting language allows.
- Java Applets are running within a *Sandbox* environment: they cannot read or write any information on the hard drive and they can only make network connections to the site from which they are downloaded. The end-user can make an exception (for signed applets) to allow access to the hard drive and network.

The Open Corpus Service in GALE allows arbitrary GAM code to be stored in the domain model, after which it is executed by the GALE Adaptation Engine (AE). This AE executes GAM event code which is arbitrary Java code that stores, retrieves and updates user model information, but that in principle can try to also do anything else. The security measures within GALE are:

- The AE runs in a Sandbox environment just like browser applets. The code has no direct access to the hard drive or the network. Its only "way out" of the Sandbox are the methods the Sandbox provides. These methods must allow the service to store and retrieve user model data.
- The only user model access that is allowed is to the user model of the user for whom the AE is executing code. This currently prevents GALE from providing "group adaptation" but it is at least "secure".

Although the adaptation engine cannot do anything "truly harmful" it does perform user model updates. And with open corpus adaptation++ the AE performs user model updates defined by possibly unknown authors. When the end-user types the URL to access a remote concept on any server through the local AE that local AE will execute whatever GAM code the unknown author has written. This code may potentially retrieve "private" user model information, and it may also destroy valuable information in the user model. This is currently a concern that is specific to GALE as GALE is the only "open corpus adaptation++ engine" we know of. GALE provides basic safety of user model information by limiting user model updates to concepts with a URI relative to the URI of the concept where the code resides. But the issue as to what should be allowed (and what not) in open corpus adaptation++ is still open in general.

4 Discussion and Conclusions

This paper presented the concept of *Open Corpus Adaptation++* where not only the *corpus* is distributed over the Web but also the *adaptation model* is distributed. This is currently just a novel feature offered by GRAPPLE's Adaptive Learning Environment GALE, and not yet widely used because the current authoring tool set GAT still does not support specifying open corpus adaptation++. The code shown in Sect. 2 is clearly not intended to be hand-written by human authors, so authoring tools will be needed.

But most importantly the paper has raised concern that open corpus adaptation++ can be potentially harmful so we should discuss what is permissible and what should be blocked for arbitrary adaptation models loaded from the Web.

Acknowledgement

We wish to thank the European Commission, project 215434 (GRAPPLE) for their financial support for this research.

References

1. Abel, F., Henze., N., Herder, E., Krause, D., Interweaving Public User Profiles on the Web, In Proceedings of UMAP 2010, User Modeling Adaptation and Personalization, LNCS 6075, pp. 16-27, Springer, 2010.
2. Brusilovsky, P., Henze, N., Open Corpus Adaptive Educational Hypermedia, in: The Adaptive Web, pp. 671-696, Springer, 2007.
3. Hendrix, M., Cristea, A.I., Design of the CAM model and authoring tool. A3H: 7th International Workshop on Authoring of Adaptive and Adaptable Hypermedia Workshop, 4th European Conference on Technology-Enhanced Learning, 2009.
4. Smits, D., De Bra, P., GALE: A Highly Extensible Adaptive Hypermedia Engine, Proc. of the ACM Conference on Hypertext and Hypermedia, Eindhoven, 2011.