Using Ontology and Data Provenance to Improve Software Processes

Humberto L. O. Dalpra[1], Gabriella C. B. Costa[2], Tássio F. M. Sirqueira[1], Regina Braga[1], Cláudia M. L. Werner[2], Fernanda Campos[1], José Maria N. David[1]

[1]UFJF – Federal University of Juiz de Fora – Department of Computer Science, Juiz de Fora – MG – Brazil.

[2]UFRJ – Federal University of Rio de Janeiro – COPPE – Systems Engineering and Computer Science Department, Rio de Janeiro – RJ –Brazil.

humbertodalpra@gmail.com, tassio@tassio.eti.br, {gabriellacbc,werner}@cos.ufrj.br, {regina.braga, fernanda.campos, jose.david}@ufjf.edu.br

Abstract. *Provenance refers to the origin of a particular object. In computational terms, provenance is a historical record of the derivation of data that can help to understand the current record. In this context, this work presents a proposal for software processes improvement using a provenance data model and an ontology. This improvement can be obtained by process data execution analysis with an approach called PROV-Process, which uses a layer for storing process provenance and an ontology based on PROV-O.*

1. Introduction

Process can be defined as a systematic approach to create a product or to perform some task [Osterweil, 1987]. Currently, many organizations are investing in the definition and improvement of their processes aiming to improve product's quality. However, the increase of process data generated makes the analysis of them more complex. It requires the use of techniques to allow proper analysis of these data, extracting records that, in fact, will contribute to process improvement. One way of analyzing this data is using provenance techniques and models.

Buneman *et al*. (2001) define data provenance as the description of the origins of a piece of data and how it is stored in a database. Thus, to capture the origin of process data, it is necessary to capture the process flow specification (prospective provenance) and process execution data (retrospective provenance), in order to have the information regarding the success, failure, delays and errors, during process execution.

Lim *et al*. (2010) state that the provenance can be captured prospectively and retrospectively. Prospective provenance captures the abstract workflow specification (or process) enabling future data derivation. Retrospective provenance captures process execution, *i.e.*, data derivation records.

To obtain the benefits of provenance, data have to be modeled, gathered, and stored for further queries [Marinho *et al*., 2012]. After the capture and storage of process provenance data, it can be used for analysis that enables process improvement (*e.g.*, shorter execution time and greater efficiency of the results). One possible way to analyze processes provenance data is through the use of ontology and the inference mechanisms provided by it, enabling the discovery of strategic information for software project managers. This paper proposes a layer for the storage of software process provenance data and the analysis of these data using an ontology. A W3C provenance model called PROV [Groth and Moreau, 2013] was used both for storage and analysis of these data.

The remainder of this paper is structured as follows: Section 2 presents related works that deal with provenance and processes. Section 3 is dedicated to describe the approach to improve software processes using an ontology called PROV-Process. The next section presents an overview of the PROV-Process ontology, which was based on PROV-O, describing the extensions made on it. Section 5 discusses the analysis of an industry software process using the PROV-Process approach and the possibilities to improve future executions of this process through the information obtained by PROV-Process ontology. Finally, conclusions are presented in Section 6.

2. Related Work

Missier *et al.* (2013) present D-PROV, an extension of PROV specification, with the aim of representing process structure, *i.e.*, to enable the storage and query using prospective provenance. An example of using D-PROV in the context of scientific workflows defined by Data ONE scientists was shown in the article. This work was used as basis to capture prospective provenance in PROV-Process approach.

Miles *et al.* (2011) propose a technique, called PRiME, to adapt application projects to interact with a provenance layer. The authors specify the steps involved in applying PRiME and analyze its effectiveness through two case studies.

Wendel *et al.* (2010) present a solution to failures in software development processes based on PRiME, the Open Provenance Model and a SOA architecture. They use Neo4j to store the data, Gremlin to query and REST web services as the connection to the tools.

Junaid *et al.* (2010) propose an approach where a provenance system intercepts the actions of users, processes and stores these actions to provide suggestions on possible future actions for the workflow project. These suggested actions are based on the actions of the current user and are calculated based on the provenance information stored.

Similar to the related work mentioned above, PROV-Process approach aims to improve future software process executions, through provenance data. However, other approaches do not use ontologies as a technique for query provenance data or use any inference mechanism, as PROV-Process approach does. Through ontology inferences, we derive strategic information to suggest software process improvement, as shown in the next sections.

3. PROV-Process Overview

PROV-Process is an approach for storage and analysis of software process provenance data in order to improve future process execution. The main objective of the approach is to identify improvements for future software process instances by using a provenance layer (comprising a database, an ontology and mechanisms to manipulate these components).

As shown in Figure 1, after the process modeling, a process instance can be created. Both the process model and the model of the generated instance are stored in

PROV-Process Database, through a prospective mechanism. After that, the process instance can be executed and the retrospective data provenance is stored through the PROV-Process approach. This storage is done using a relational database, which has been modeled using PROV-DM specification [Moreau and Missier, 2013].

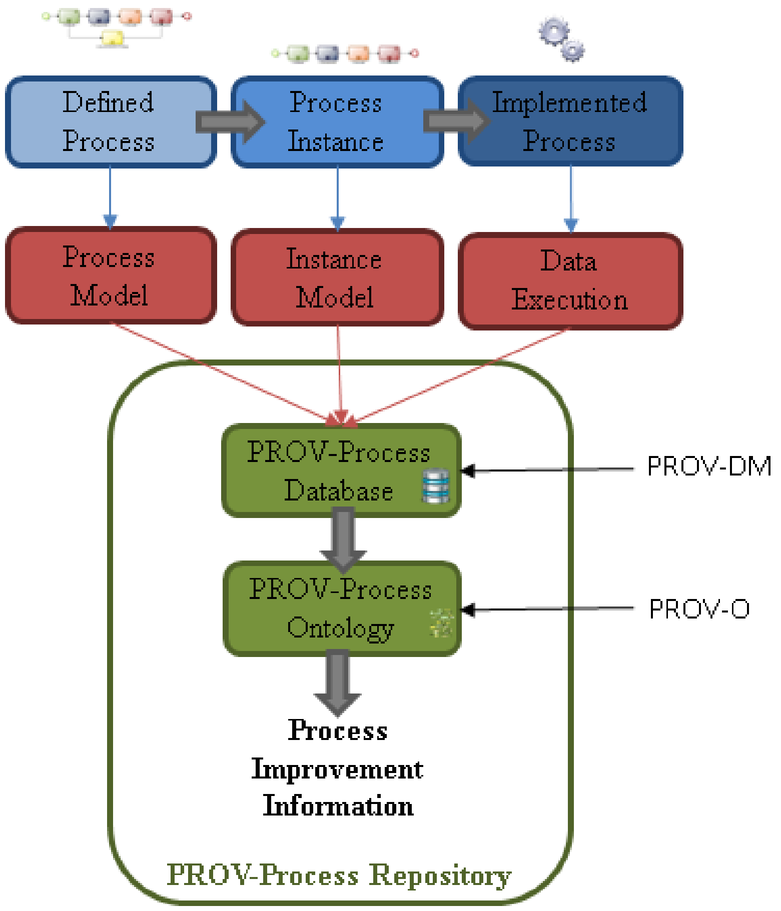


Figure 1: PROV-Process Approach

PROV-DM types and relations are organized according to six components. PROV-Process Database implements all these components using a relational database. Figure 2 shows, for example, tables of the first component, which comprise entities, activities and their interrelations: Used (Usage), WasGeneratedBy (Generation), WasStartedBy (Start), WasEndedBy (End), WasInvalidatedBy (Invalidation), and WasInformedBy (Communication).

All the data stored in the PROV-Process relational database are exported to the PROV-Process ontology. This ontology is described in details in next section.

4. PROV-Process Ontology

Ontology research has become more widespread in Computer Science community. Although the term has been limited to the philosophy sphere in the past, it has earned specific roles in Artificial Intelligence, Computational Linguistics and Databases [Guarino, 1998].

PROV-Process Ontology was developed from the PROV-O ontology [Belhajjame *et al*., 2013], which was defined based on PROV-DM data model. PROV-O defines the vertices of PROV (Agent, Entity and Activity) as classes and uses object properties for the interrelations representation. The core classes and properties from PROV-O are shown in Figure 3.

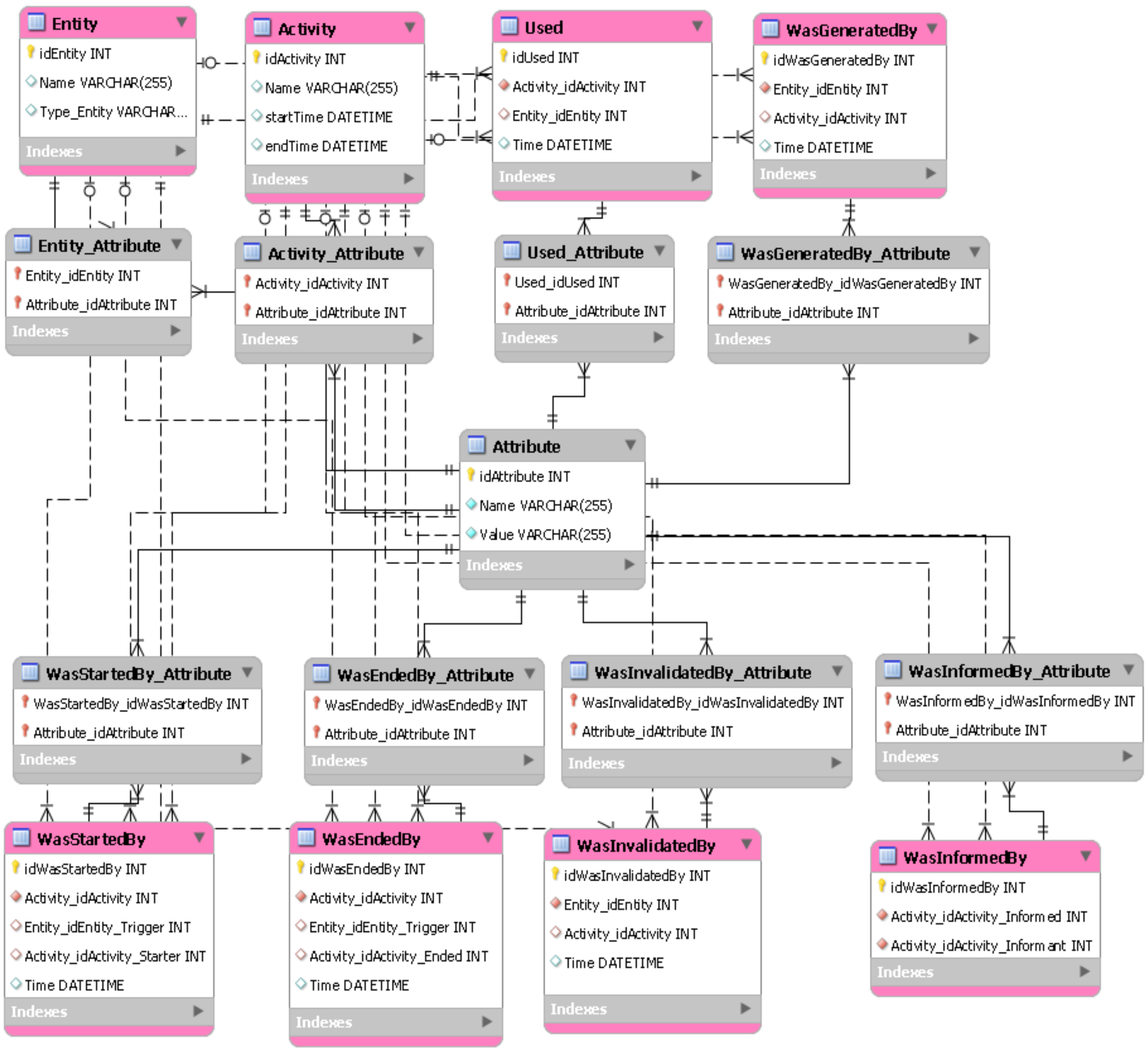


Figure 2: Part of PROV-Process Database

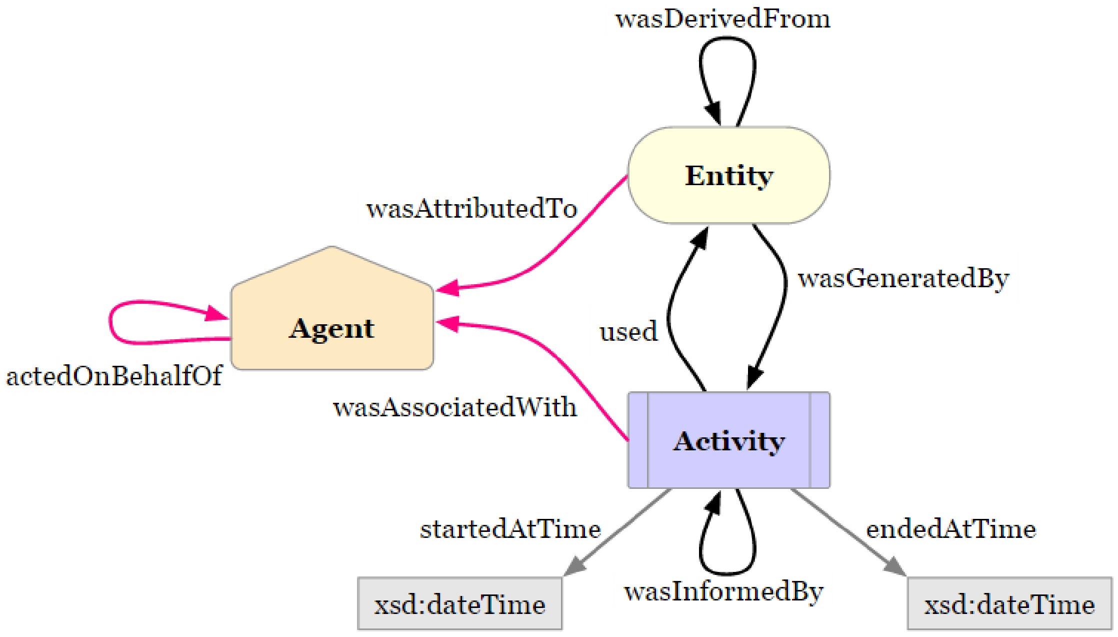


Figure 3: PROV-O: Core Classes and Properties [Belhajjame *et al*., 2013]

Classes and properties in PROV-O can be used directly to represent provenance information or one can specialize them for modeling specific applications. Thus, PROV-O can also be specialized to create new classes and properties to model provenance information for different domains and applications. Based on this, we create some new properties on PROV-O (generating PROV-Process Ontology), in order to adapt it to the software process domain and to allow the inference of new information to improve software processes. Examples of these properties are presented in the following.

A group of rules (using Property Chains) was added in PROV-O in the 'wasAssociatedWith' data property:

1. **used** o **wasAttributedTo**
2. **wasStartedBy** o **wasAttributedTo**
3. **wasEndedBy** o **wasAttributedTo**

These rules state that, as show in Figure 4, if an activity used, was stated by or was ended by an entity and that entity was assigned to an agent, we can infer that an activity is associated with an agent.

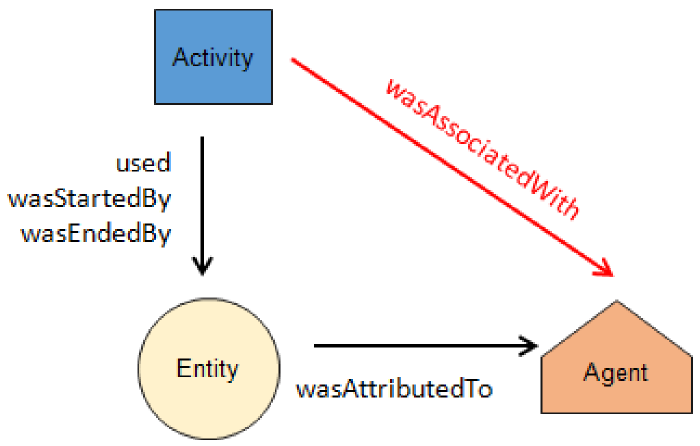


Figure 4: wasAssociatedWith properties chains

In the PROV-O, a data property called *processInstanceId* that corresponds to the generated/executed instance identifier from the main process was also inserted.

Finally, it should be noted that all records, called *Attributes* in PROV-Process database, must be exported to the PROV-Process Ontology as new data properties with their respective value.

5. Evaluation

In order to evaluate the applicability of the ontology of PROV-Process to software process, the approach was applied to a process from a Brazilian software development company [Ceosoftware, 2015]. A flow model shown in Figure 5 was created based on the specifications of this process.

To do this evaluation, real data execution of the process expressed in Figure 5 was analyzed. Thus, retrospective provenance of this process instances was stored using the PROV-Process relational database. It should be noted, however, that the execution data of the whole process were not provided by the company, but just a part of it.

In this work, 10 process execution instances, which have been fully completed, were analyzed. Regarding the obtained data, the following were used:

RDM[1] (change request) number;

- Information if an RDM was created from a previous RDM;
- Date and time of RDM opening;
- Type of RDM;
- Responsible for opening the RDM (Origin);
- Changed modules and components during the deployment task;
- Team responsible for implementation of the solution;

[1] RDM is an acronym for ‘change request’, in Portuguese, used by the company which provided the data for this research. It means a registration opened by support, client or commercial department, to make changes / adjustments in software system.

- Situation of RDM;
- Date and time of RDM completion.

These process execution data were obtained through a spreadsheet sent by the company responsible for the project[2].

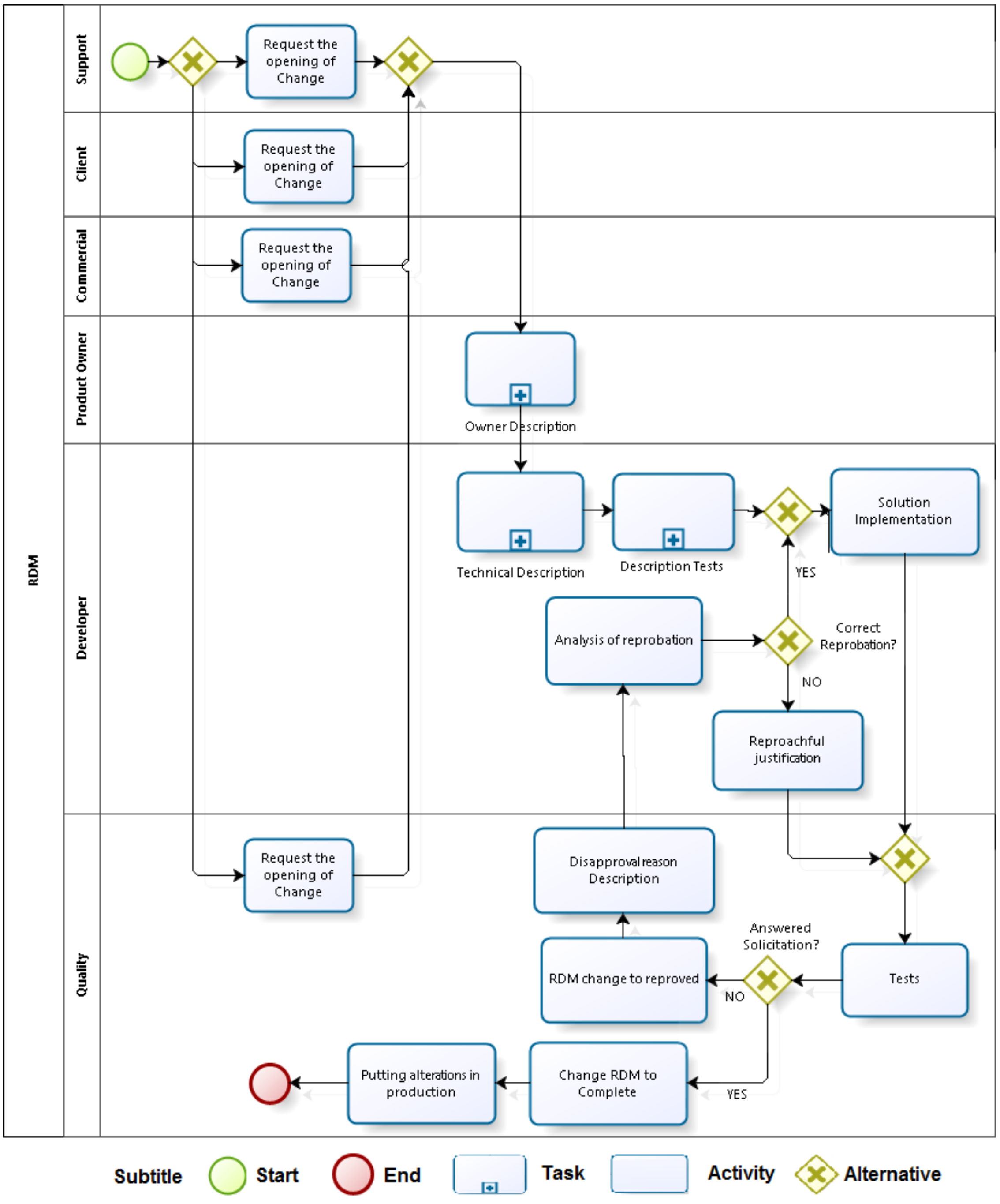


Figure 5: Process to manage requests and changes in software

[2] All the execution data used for the implementation of this assessment can be found at this link http://gabriellacastro.com.br/dsc/ex1/ex1.xlsx . Each row of this table represents a distinct execution of the process.

Table 1: Data execution example – Part 1

RDM Number	Outspread	Opening Date	Opening Time	Type	Origin
30006	0	10/03/2013	14:54:00	Module liberation	Client
30006	1	06/11/2014	17:18:00	Module liberation	Client

Table 2: Data execution example – Part 1

RDM Number	RDM Module	Module	Component	Team	Closed Date	Closed Time
30006	Financial	DLL - ERP PDA	clsValidacao	VB6	10/03/2013	22:06:00
30006	Financial	DLL - ERP PDA	clsValidacao	VB6	06/12/2014	10:41:00

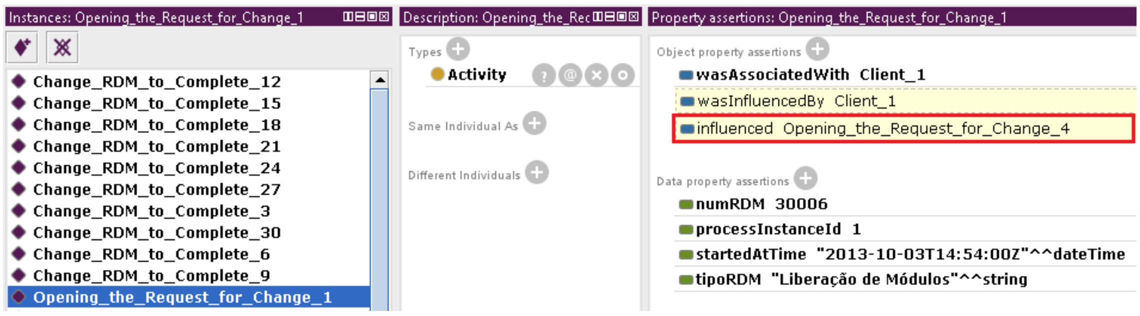
The obtained data (examples about these data can be seen in Tables 1 and 2, where the dates are using the format MM/DD/YYYY) were imported to the PROV-Process database according to the following criteria:

- For all executions whose data were analyzed, three records were set in *Activity* table, with their names:
  - Opening the Request for Change;
  - Solution Implementation;
  - Change RDM to Complete.
- The RDM number was inserted as an attribute of each of the above activities, by using *Attribute* and *Activity_Attribute* tables.
- If a particular instance of execution corresponds to the unfolding of a previous RDM, a record in *WasInfomedBy* table was created.
- Date and time of RDM open were included using the *startTime* attribute of the activity Opening the Request for Change.
- RDM type was inserted as an attribute of the Opening the Request for Change activity using *Attribute* and *Activity_Attribute* tables.
- The role responsible for the Opening the Request for Change activity was inserted in *Agent* table, using the *name* field and the Person type.
- Relationship between Opening the Request for Change activity and the responsible for the same activity were inserted as records of the *WasAssociatedWith* table.
- Values as module, RDM module and component were included as records using *Entity* table and were associated with the Solution Implementation activity by creating records in *Used* table.
- Values as module, RDM module and component, included as records using *Entity* table, have been associated with agents who manipulated it by creating records in *WasAttributedTo* table.
- Role responsible for Solution Implementation activity was inserted in *Agent* table, using the *name* field and the Person type.

- Relationship between Solution Implementation activity and the responsible for the same activity were inserted as records in *WasAssociatedWith* table.
- Date and time of RDM completion were inserted using the *endTime* attribute of the Change RDM to Complete activity.
- As in the flow model (Figure 5) the role responsible for the Change RDM to Complete activity is the Quality Team. This role was inserted as record in the *Agent* table and was associated with this task by inserting a record in the *wasAssociatedWith* table.
- In order to identify which instance of the process execution a particular activity is associated with, a related attribute called *processInstanceId* was added to all activities by using the *Attribute* and *Activity_Attribute* tables.

After inserting the process execution data in the PROV-Process relational database, all the data were entered as individuals and their relationship in PROV-Process Ontology[3]. From this point, through the ontology inference engine, the derivation of strategic information was possible. As examples of information inferred from retrospective provenance data of this process, we can highlight four types:

1) Activities that influenced the generation of other activities, that is, as can be seen in red mark in Figure 6, Opening the Request for Change (id = 1) influenced Opening the Request for Change (id = 4). The same information was also inferred for the tasks of the same type with the ids 7, 13 and 19.



Instances: Opening_the_Request_for_Change_1

Change_RDM_to_Complete_12
Change_RDM_to_Complete_15
Change_RDM_to_Complete_18
Change_RDM_to_Complete_21
Change_RDM_to_Complete_24
Change_RDM_to_Complete_27
Change_RDM_to_Complete_3
Change_RDM_to_Complete_30
Change_RDM_to_Complete_6
Change_RDM_to_Complete_9
Opening_the_Request_for_Change_1

Description: Opening_the_Rec

Types
Activity
Same Individual As
Different Individuals

Property assertions: Opening_the_Request_for_Change_1

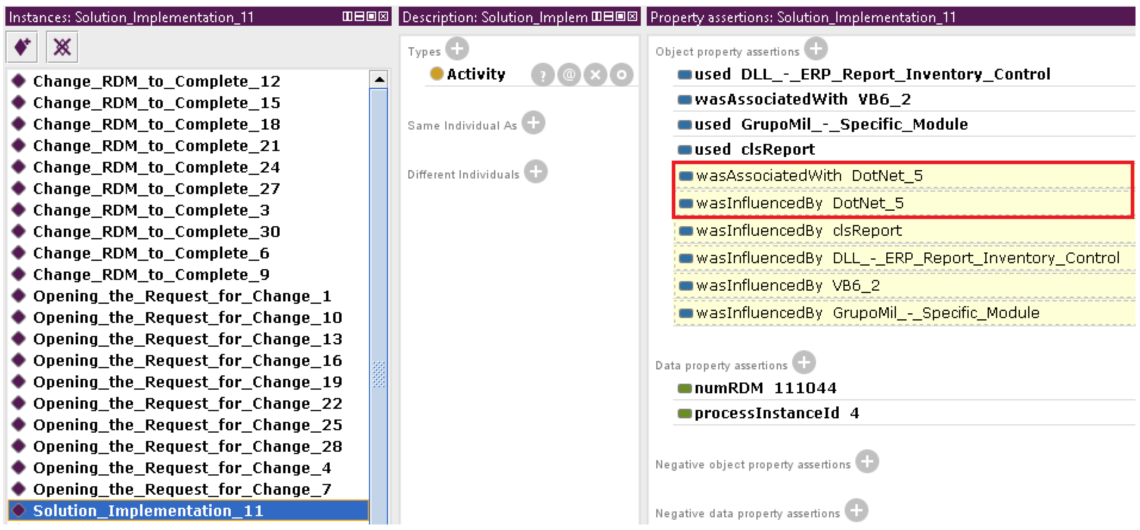
Object property assertions
wasAssociatedWith Client_1
wasInfluencedBy Client_1
influenced Opening_the_Request_for_Change_4

Data property assertions
numRDM 30006
processInstanceId 1
startedAtTime "2013-10-03T14:54:00Z"^^dateTime
tipoRDM "Liberação de Módulos"^^string

Figure 6: Activities that influenced the generation of other activities

2) Agents that could be associated with the Solution Implementation activity, considering that they already handled the artifacts involved in this activity in any other execution of the process. Figure 7 shows, for example, that Solution Implementation activity (id = 11) was influenced by DotNet agent (id = 5), given that this agent handled common artifacts to this activity in other instances of this process. The same type of information (agents that could be associated with the Solution deployment task) also occurs for Solution Implementation activity with ids equal to 8, 20, 23, 26 and 29.

[3] The generated ontology with all the individuals can be found at this link http://gabriellacastro.com.br/dsc/ex1/ex1-english.owl .


Instances: Solution_Implementation_11

Change_RDM_to_Complete_12
Change_RDM_to_Complete_15
Change_RDM_to_Complete_18
Change_RDM_to_Complete_21
Change_RDM_to_Complete_24
Change_RDM_to_Complete_27
Change_RDM_to_Complete_3
Change_RDM_to_Complete_30
Change_RDM_to_Complete_6
Change_RDM_to_Complete_9
Opening_the_Request_for_Change_1
Opening_the_Request_for_Change_10
Opening_the_Request_for_Change_13
Opening_the_Request_for_Change_16
Opening_the_Request_for_Change_19
Opening_the_Request_for_Change_22
Opening_the_Request_for_Change_25
Opening_the_Request_for_Change_28
Opening_the_Request_for_Change_4
Opening_the_Request_for_Change_7
Solution_Implementation_11

Description: Solution_Implem

Types
Activity

Same Individual As

Different Individuals

Property assertions: Solution_Implementation_11

Object property assertions
used DLL_-_ERP_Report_Inventory_Control
wasAssociatedWith VB6_2
used GrupoMil_-_Specific_Module
used clsReport
wasAssociatedWith DotNet_5
wasInfluencedBy DotNet_5
wasInfluencedBy clsReport
wasInfluencedBy DLL_-_ERP_Report_Inventory_Control
wasInfluencedBy VB6_2
wasInfluencedBy GrupoMil_-_Specific_Module

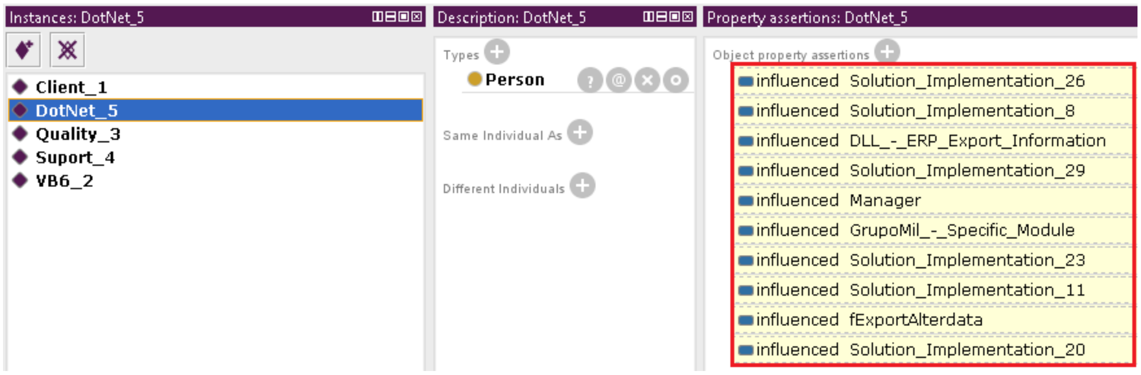
Data property assertions
numRDM 111044
processInstanceId 4

Negative object property assertions

Negative data property assertions

Figura 7: Agents that influenced an activity

3) A list of all activities in which an agent was involved, as well as the artifacts (entities) handled by her/him, as can be seen in Figure 8. Although this type of information can be obtained through queries on PROV-Process relational database, using the ontology and inference engine, this information can be obtained more easily (with a simple SPARQL query).


Instances: DotNet_5

Client_1
DotNet_5
Quality_3
Suport_4
VB6_2

Description: DotNet_5

Types
Person

Same Individual As

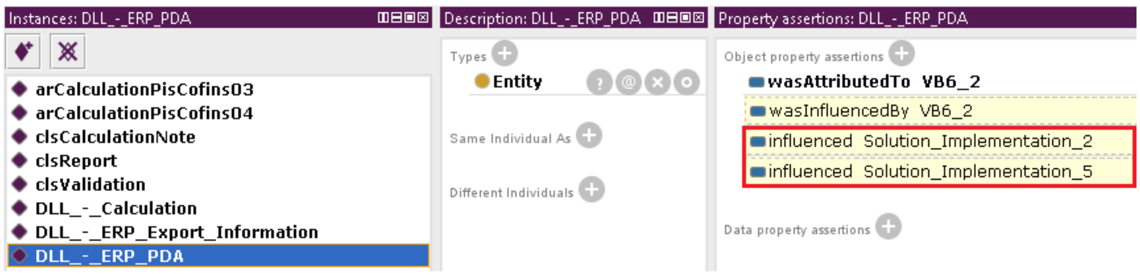
Different Individuals

Property assertions: DotNet_5

Object property assertions
influenced Solution_Implementation_26
influenced Solution_Implementation_8
influenced DLL_-_ERP_Export_Information
influenced Solution_Implementation_29
influenced Manager
influenced GrupoMil_-_Specific_Module
influenced Solution_Implementation_23
influenced Solution_Implementation_11
influenced fExportAlterdata
influenced Solution_Implementation_20

Figure 8: Activities and agents handled by DotNet agent

4) A list of all activities where an artifact (entity) was consumed, as can be seen in Figure 9. Although this type of information can be obtained through queries on PROV-Process relational database, using the ontology and inference engine, this information may be obtained more easily (with a simple SPARQL query).


Instances: DLL_-_ERP_PDA

- arCalculationPisCofins03
- arCalculationPisCofins04
- clsCalculationNote
- clsReport
- clsValidation
- DLL_-_Calculation
- DLL_-_ERP_Export_Information
- DLL_-_ERP_PDA

Description: DLL_-_ERP_PDA

Types
Entity

Same Individual As

Different Individuals

Property assertions: DLL_-_ERP_PDA

Object property assertions
wasAttributedTo VB6_2
wasInfluencedBy VB6_2
influenced Solution_Implementation_2
influenced Solution_Implementation_5

Data property assertions

Figura 9: Activities where an artifact (entity) was consumed

Information inferred from the use of ontology proposed by the PROV-Process approach could help to improve process performance offering to the project manager information acquired at the time of the instantiation of a new process. This information might suggest, for example, the most appropriate agents and artifacts to be handled first, according to the type of problem reported / reason for opening the RDM.

6. Conclusion

This paper presented an approach, called PROV-Process, that obtains strategic information to the project manager enabling her/him to take decisions that can improve process performance. Therefore, this approach presents the advantages of using data provenance coupled with ontology. Through the use of ontology, it is possible to detect: (1) activities that influenced the generation of other activities; (2) agents that could be associated with the solution of the deployment task, considering that they already handled the artifacts involved in this task in any other execution of the process; (3) A list of activities in which an agent was involved, as well as the artifacts (entities) handled by her/him. No metric had be used for comparison of results.

Process data execution (retrospective provenance of the software process) are stored in a relational database, modeled based on PROV-DM specification. As a result, its data feed an ontology, created from the PROV-O model. With this and using an inference machine, one can infer new information about the process.

To evaluate the PROV-Process approach, it was applied to a real industry software process.

As threats to validity, we can cite:

- The partner company did not inform all process performance data, and only made available a spreadsheet with some of these data. This lack of detail directly impacts on a greater specificity in the results.
- Data obtained from the partner company did not include information about the actors who, in fact, performed the activity. They informed only the team that performed a certain activity.

Currently, we are working in the following improvements: (1) Implementing new rules indicating other actions that can help to improve processes; (2) Applying the approach to other real case studies.

References

Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J. (2013) "PROV-O: The PROV Ontology". Available in <http://www.w3.org/TR/prov-o/>. Accessed in July 2015.

Buneman, P., Khanna, S. and Tan, W. C. (2001) "Why and where: A characterization of data provenance". In: 8th International Conference on Database Theory, London. pp. 4-6.

Ceosoftware. (2015) "Soluções criativas e inovadoras". Available in <http://www.ceosoftware.com.br/>. Accessed in July 2015 (in Portuguese).

Groth, P., Moreau, L. (2013) "PROV - Overview". Available in <http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/>. Accessed in July 2015.

Guarino, N. (1998) "Formal ontology in information systems" In: Proceedings of the first international conference (FOIS'98), Trento, Italy (Vol. 46). IOS press, pp. 3-15.

Junaid, M. M., Berger, M., Vitvar, T., Plankensteiner, K., Fahringer, T. (2009) "Workflow composition through design suggestions using design-time provenance information". In: E-Science Workshops, 2009 5th IEEE International Conference on. IEEE. pp. 110-117.

Lim, C., Lu, S., Chebotko, A., Fotouhi, F. (2010) "Prospective and Retrospective Provenance Collection in Scientific Workflow Environments". In Proceedings of the 2010 IEEE International Conference on Services Computing (SCC '10). IEEE Computer Society, Washington, DC, USA, pp. 449-456.

Marinho, A., Murta, L., Werner, C., Braganholo, V., Cruz, S. M. S. D., Ogasawara, E., Mattoso, M. (2012) "ProvManager: a provenance management system for scientific workflows". Concurrency and Computation: Practice and Experience, v. 24, n. 13, pp. 1513-1530.

Miles, S., Groth, P., Munroe, S., Moreau, L. (2011) "PrIMe: A methodology for developing provenance-aware applications". ACM Transactions on Software Engineering and Methodology (TOSEM), v.20 n.3, pp.1-42.

Missier, P., Dey, S. C., Belhajjame, K., Cuevas-Vicenttín, V., Ludäscher, B. (2013) "D-PROV: extending the PROV provenance model with workflow structure". In: Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance (TaPP). USENIX Association, Berkeley, CA, USA, Article 9, pp. 1-7.

Moreau, L., Missier, P. (2013). "Prov-dm: The prov data model". Available in <http://www.w3.org/TR/2013/REC-prov-dm-20130430/>. Accessed in July 2015.

Osterweil, L. (1987) "Software processes are software too". In Proceedings of the 9th international conference on Software Engineering. IEEE Computer Society Press, pp. 2-13.

Wendel, H., Kunde, M., Schreiber, A. (2010). "Provenance of software development processes". In Provenance and Annotation of Data and Processes, v. 6378 of Lecture Notes in Computer Science, pp. 59-63. Springer Berlin Heidelberg.