SPPV: Visualizing Software Process Provenance Data

Gabriella C. B. Costa[1], Marcelo Schots[1], Weiner E. B. Oliveira[2], Humberto L. O. Dalpra[2], Cláudia M. L. Werner[1], Regina Braga[2], José Maria N. David[2], Marcos A. Miguel[2], Victor Ströele[2], Fernanda Campos[2]

[1]COPPE - Systems Engineering and Computer Science Department
UFRJ - Federal University of Rio de Janeiro
21945-970 - Rio de Janeiro - RJ - Brazil

[2]Computer Science Department
UFJF - Federal University of Juiz de Fora
36036-900 - Juiz de Fora - MG - Brazil

{gabriellacbc, schots, werner}@cos.ufrj.br, {woliveira82, humbertodalpra}@gmail.com, {regina.braga, jose.david, fernanda.campos}@ufjf.edu.br, {marcos.miguel, victor.stroele}@ice.ufjf.br

Abstract. *Provenance data can provide implicit and strategic information for process improvement, helping to establish potential causes for process success, failure, delays, among others. However, when it comes to software processes, the large amount of execution data generated makes their analysis complex and arduous, requiring proper ways to store and represent provenance information. In this sense, this paper presents the application of a goal-oriented process to build a visualization tool geared to provenance data. The suitability of the visual attributes mapped was preliminarily assessed through an exploratory analysis with stakeholders from two software development companies, using their own process provenance data.*

1. Introduction

A software process can be defined as a set of activities, methods, practices and transformations that people use to develop and maintain software and associated products. A standard software process encompasses the essential process assets: activities, artifacts, resources, and procedures [Falbo and Bertollo, 2005].

Provenance "is a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing" [Belhajjame *et al*. 2013]. Data provenance can be used in the context of software process development to provide additional information about it. For example, during process modeling and execution, data provenance can be captured to establish process success, failure, delays and errors [Wendel *et al*. 2010].

Assuming that software development organizations in their intra-organizational context perform similar processes, the knowledge acquired in previous executions of these processes can be reused to establish better policies to adopt in future projects, thus supporting continuous process improvement. This knowledge can also be acquired by investigating the data generated during process executions. However, the increase of

process data generated during this execution makes the data analysis more complex. Thus, it requires techniques that allow a proper examination of these data, extracting records and facts that will actually contribute to process improvement and choose a proper representation for these data to enable their exploration and understanding.

Most software process managers are focused on process monitoring and improvement, and they do not have advanced knowledge about models, methods, and techniques for software process data analysis. For analyzing software process data, the use of provenance techniques and models and appropriate data visualizations can be of great help. Our main contribution is focused in proposing an appropriate data provenance visualization in a specific domain (software development process), in order to facilitate the understanding about these data and to support software process improvement. We did not find any related contribution targeted to software process provenance data and a way to visualize them. Generic provenance visualization tools do not use a familiar notation to process managers and often provided data that do not offer any meaning for them. The approach presented in this paper aims to assist process managers in (i) understanding software process provenance data, and (ii) using them to support decision-making on the analyzed process.

The remainder of this paper is structured as follows. The next section provides an overview of the PROV-Process approach [Dalpra *et al*. 2015], used as the visualization backend for software process provenance data handling and storage. Section 3 discusses other works related to provenance visualization. Section 4 describes how the visualization attributes were defined, using a goal-oriented process [Schots and Werner 2015]. The SPPV (Software Process Provenance Visualization) tool, developed to concretize the proposed approach, is presented in Section 5 with an exploratory analysis, using software process provenance data from two Brazilian software development companies. Finally, the conclusions are presented in Section 6.

2. The PROV-Process Approach

The PROV-Process approach was used for structuring and storing the software process provenance data to be visualized. It is part of the iSPuP (improving Software Process using Provenance) approach [Costa *et al*. 2016] and consists of a specified architecture for capturing, storing and analyzing processes provenance data, using the PROV model [Moreau *et al*. 2015]. PROV-Process' database was modeled and implemented based on PROV-DM [Belhajjame *et al*. 2013], which suggests three vertices to represent *entities*, *activities* and *agents*, including causal relationships between them, such as *wasGeneratedBy*, *wasStartedBy*, *wasEndedBy*, *wasInvalidatedBy*, *wasDerivedFrom*, *alternateOf*, *specializationOf*, *used* and *hadMember*. In this approach, the *entities* represent the artifacts of software process, *activities* are used with the same concept of software process activities, and *agents* represent software process resources.

In addition to allowing the storage of provenance data, PROV-Process offers an interface to build an OWL (Ontology Web Language) file with the captured provenance data of a software process using an extension of PROV-O ontology [Lebo *et al*. 2013], named PROV-Process Ontology. Figure 1 shows an example of an activity instance (called *Solution_Implementation_14*) represented in PROV-Process Ontology. As it can be seen, this activity used four entities and was associated with the agent *VB6_2* (as

illustrated by the used and *wasAssociatedWith* relationships). It can be noticed that, as the amount of data increases, the analysis of them in an ontology tool (such as Protégé) becomes a non-trivial task. It is known that software became bigger over the years, generating large amounts of daily information. Thus, the purpose of using software visualization applied to provenance data is to help understanding the processed information in order to improve the efficiency of software development processes.

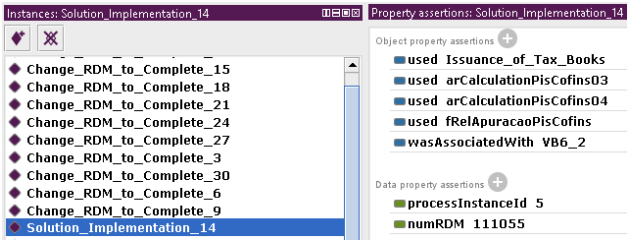


Figure 1. Activity example represented in the ontology tool.

3. Related Work

NoWorkflow [Murta *et al*. 2014] captures provenance of experiment scripts and includes a graph-based visualization mechanism. This approach only captures provenance of Python scripts and is not focused on capturing, storing, and analyzing software process provenance data. Besides, it does not use mechanisms to emphasize a given item (for example, an agent), as SPPV does.

Chen and Plale (2015) use visualization techniques to support real time analysis of large provenance graphs, applying it to the provenance captured from the network layers of large-scale E-Science distributed applications. Unlike this proposal, our work combines the use of provenance data, ontology and visualization attributes, in order to facilitate project managers in understanding the analyzed software process.

Some tools such as GraphViz [Ellson *et al*. 2001] and PROV-O-Viz [Hoekstra and Groth 2014] deal with ontology visualizations as graphs. Differently from GraphViz, SPPV focuses on ontology individuals, and is targeted to experts in software process management, who does not necessarily have much knowledge in ontology. Besides, SPPV uses BPMN, a more familiar notation to process managers. PROV-O-Viz, in turn, is a web-based visualization tool for PROV-based provenance traces collected from various sources, including ontologies, which leverages diagrams to reflect the flow of information through activities. Since our focus is not at analyzing the flow of information, PROV-O-Viz becomes unsuitable for our specific goals.

Although Wendel *et al*. (2010) present an approach that handles provenance and software development processes, we did not find contributions targeted to software process provenance data and a way to visualize them. SPPV uses an explicit rationale to identify appropriate visualization attributes [Schots and Werner 2015], and was subjected to an exploratory analysis with real data from two Brazilian companies.

4. Mapping Process Provenance Data and Visualization

When creating a visualization tool, there are specific goals to be achieved. The challenge lies on transforming software process provenance data into a corresponding visualization. To do this, we used the staged process proposed by Schots and Werner

(2015) for mapping managerial needs to visualization attributes. To develop the proposed provenance visualization system, the meet-in-the-middle strategy was chosen.

4.1. Mapping Goals and Questions

In this study, two goals were previously defined: (**G1**) *Analyze whether a process agent, who is a company employee engaged in the process tasks, is overloaded in a given process*; and (**G2**) *Analyze how often process requests are started by clients*. The first goal (**G1**) aims to assist the process manager in making decisions regarding employees who are overloaded or idle in the process, being able to perform a different distribution of tasks in the process. The second goal (**G2**) aims to assist in analyzing client demands, considering that a process generates a software product that is consumed by a client.

Considering these goals, the following questions were raised: (**Qa**) *How many entities/artifacts were manipulated by the agent?* (related to **G1**); (**Qb**) *How often are these entities manipulated by the agent over time?* (related to **G1**); (**Qc**) *How many activities/tasks were done by the agent?* (related to **G1**); (**Qd**) *How often are these activities done by the agent over time?* (related to **G1**); (**Qe**): *How many activities/tasks, related to new requests, were started by the client?* (related to **G2**). Although tabular or list-based views can help answering these questions, the context in which the entities, agents and activities were involved and, most importantly, the other relations between them, cannot be interpreted as easily as by using a visualization.

4.2. Mapping Questions and Tasks

The tasks associated with these questions are: (**Ta**) *Analyze the amount of activities/ tasks performed by each agent* (related to **Qc** and **Qd**); (**Tb**) *Analyze the amount of entities/artifacts manipulated by each agent* (related to **Qa** and **Qb**); (**Tc**) *Analyze the amount of activities/tasks related to new requests started by each client* (related to **Qe**).

4.3. Mapping Tasks and Data

The data required to support the tasks are: (i) **agent information**: agent name (**Ta**, **Tb**, **Tc**) and type: Organization, Person or Software Agent (whose roles can be a client, a developer, or a software to automate a certain task); (ii) **activity information**: activity name (**Ta**, **Tc**) and its respective data properties; (iii) **entity information**: entity name (**Tb**); (iv) **relation information**: relation name, source, target, type (asserted or inferred[1]), and amount of relations between the same source and target (**Ta**, **Tb**, **Tc**).

4.4. Mapping Data and Visualization

One of PROV requirements [Moreau *et al*. 2015] is to provide a single layout convention used throughout PROV specifications. This convention uses blue rectangles, yellow ellipses, and orange pentagons for activities, entities, and agents, respectively. This layout convention was adopted in our mapping, but since process experts are not familiar with the conventional layout used by PROV, the developed tool also enables the visualization of processes provenance data using the BPMN 2.0 notation [OMG,

[1] An asserted relation comes from the original process data within a knowledge base, while an inferred relation represents additional process data that some ontology reasoned provided/inferred from the asserted data.

2011]. Table 1 depicts the mapping of software process provenance data to visual attributes. In addition, SPPV also highlights elements and relationships associated to an element when hovering the mouse on it. It also provides filters to allow the selection of specific elements of the ontology (i.e., an activity, an entity, or an agent).

Table 1. Data-to-Visualization Mapping.

Visual Attribute	Data	Value	Description
shape / icon	Agent	pentagon OR person icon*	Represents an agent in PROV or a role in BPMN.
	Activity	rectangle	Represents an activity in PROV or a task in BPMN.
	Entity	ellipse OR paper icon	Represents an entity in PROV notation or an artifact in BPMN.
	Used	arrow	Represents the use of an entity by an activity.
	wasAssociatedWith		Represents an association between an activity and an agent.
	wasAttributedTo		Represents the assignment of an entity to an agent.
	Influenced		Represents an abstract relation of influence.
	wasInfluencedBy		
color	Agent	orange	Used to identify the type of vertex (as the agent symbol does not exist in BPMN, we created an icon to represent the respective nodes.)
	Activity	blue	
	Entity	yellow	
	relation type	- black, for asserted relations - red, for inferred relations - olive, for inferred and asserted relations	The use of ontology allows inferences to be made using provenance data. Then, the arrows in black display provenance data and the arrows in red display inferred data.
transparency	amount of relations in which the vertex (agent, activity, or entity) is part	- high: for 1 relation - medium: for 2 or 3 relations -low: for 4 or more relations	According to the vertex brightness, it will be easier to identify the one that has more relations than others.
width	amount of relations between the same vertex source and target	- thick: for 4 or more relations - medium: for 2 or 3 relations - slim: for 1 relation	When there is more than one relation between the same source and target, the arrow width will facilitate to identify this amount of relations.

*As the agent symbol does not exist in BPMN, we created an icon to represent the respective nodes.

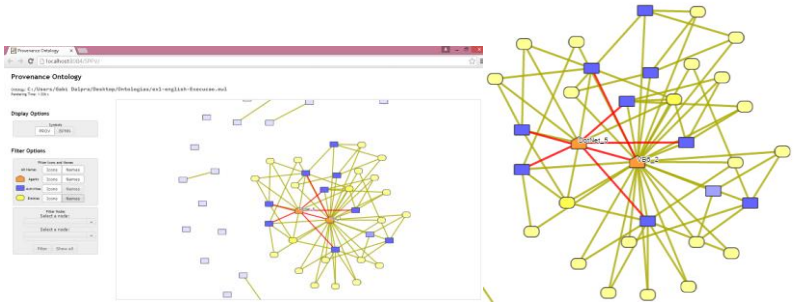
5. The Software Process Provenance Visualization Tool (SPPV) in Action

The SPPV tool loads the provenance data from ontology as can be seen in Figures 2 to 5. An exploratory study was performed using SPPV to analyze the data from two real industry processes. One process is used to manage change requests in a 19-years-old Brazilian software company (referred to as company α) that deals with business management software. The other process involves the implementation of new features and error handling in an Enterprise Resource Planning project from a small Brazilian software development company (referred to as company β). Both companies provided a log from the executed process instances through spreadsheets, transformed in a CSV file to be used as input by PROV-Process. The data were then anonymized and stored in the relational database. Ten process execution instances from each company were used. The goal of the exploratory study was to evaluate if the goals established in Subsection 4.1 could be achieved with the visualization attributes implemented in SPPV.

The visualization of the provenance data from the first process is shown in Figures 2 and 3. By executing the proposed tasks using SPPV, the goals G1 and G2 were met as follows: (**G1)** Process agent *VB6_2* is overloaded, while agent *DotNet_5* performed few tasks (Figure 2 (b)); (**G2)** Client *Client_1* opened a much higher number of requests when compared with *Quality_3* and *Support_4* (Figure 3(b)).

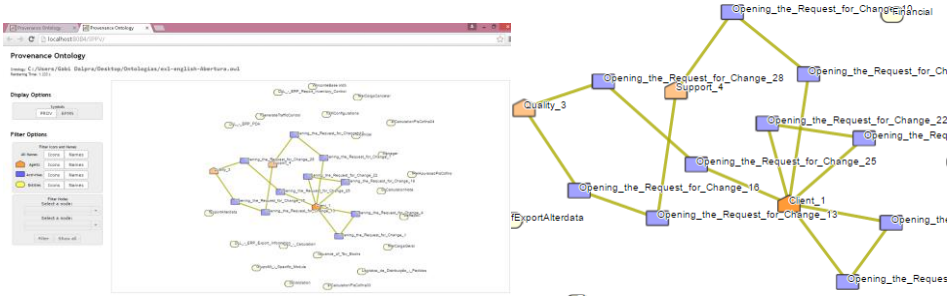
The visualization of the provenance data from the second process is shown in Figures 4 and 5. The goals set in **G1** and **G2** were achieved with SPPV as follows: (**G1)**: Process agent *Developer_4* is overloaded, while agents *Developer_6*, *Developer_7*, and

Developer_8 performed only one task (Figure 4(b)). (**G2**) Client *Client_E* did a much higher number of requests when compared to all the other clients (Figure 5(b)).



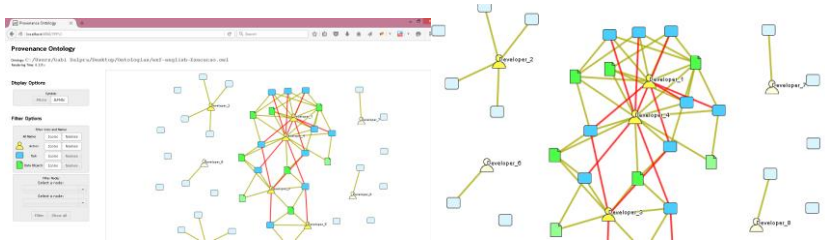
(a) (b)

Figure 2. Process 1 - visualization for **G1**: process agent overloaded.



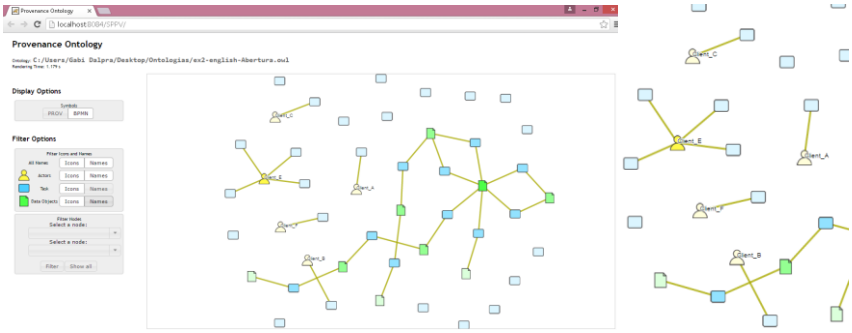
(a) (b)

Figure 3. Process 1 - visualization for **G2**: process requests started.



(a) (b)

Figure 4. Process 2 - visualization for **G1:** process agent overloaded.



(a) (b)

Figure 5. Process 2 - visualization for **G2**: process requests started.

A preliminary evaluation about the proposed tool was done through interviews composed by some specific questions with people who participate in the processes. From company α, the interviewee was involved in the process for five years as a

software developer. From company β, the interviewee was the process manager, who was involved in the process since its inception (for more than ten years).

When participants were asked about which person performs more tasks in their company processes in the last ten instances, **Rα** stated that "a way to do this is by executing a single query in our database," while **Rβ** affirmed, "the company uses Mantis, and I can find the information on it." Participants were also asked to identify the agents who were most overloaded, based on SPPV. Both answered correctly. They were then asked if they could see something in this visualization, besides the overload of tasks. **Rα** mentioned "the relation between the agents and system artifacts" and **Rβ** said, "tasks that involved more than one person for their resolution caught my attention. This may indicate that the task was complex or required several revisions until the process was completed." The last question about **G1** was: "By knowing that an agent is overloaded with tasks, what actions would you take?" **Rα** answered, "one possible action is to analyze who has done few tasks in the process and try to delegate the tasks that have been made by the person who is overloaded," while **Rβ** said that "this type of information may indicate that (...) either the person who is apparently overloaded needs his/her responsibilities to be reviewed, or maybe this person is actually more efficient than the rest of the team and, thus, has done more tasks than the others." These answers allow concluding that the definition of what to do with this information depends on the prior knowledge of the person who is performing the analysis.

Related to **G2**, participants were asked about which client performs more requests, without using SPPV. **Rα** said that he could obtain this information through the sales sector contact, and **Rβ** said that he cannot obtain this type of information. Both participants correctly identified clients who performs more requests. By seeing a client who performs more requests than others, **Rα** said that he would "analyze why this is happening," and **Rβ** answered that "this information is important because we can analyze if the client who has opened more requests in last executions has a system version that presents many problems, or if his/her requirements are related to new implementations to this system; if the latter is true, the company's commercial sector needs to be contacted for a possible re-examination of this client's service pack."

When participants were asked about what they liked in SPPV and what they would change, **Rα** answered that "the relations between nodes are not clear (...); besides, the tool does not need all types of symbols (BPMN and PROV). One is enough." **Rβ** said that "the SPPV visualization using the proposed graphs and symbols is interesting. The information discovered using the visualization might be useful to increase the company's revenues"; he added, "the number of relations of each edge should be displayed. One visualization – BPMN or PROV – is enough, as both show the same things. A filter to select actors from a specific team would be interesting, too."

6. Conclusion

This paper described the usage of a goal-oriented mapping process for planning SPPV, a tool for visualizing software process provenance data. An exploratory analysis using two real-world industry processes was carried out, with a positive feedback from the process experts. Although the exploratory analysis was performed in a real world context, it cannot be generalized to the context of other industry software processes.

As future work, additional goals and attributes (e.g., time spent on activities) based on provenance data can be defined with their respective visualization(s). We also intend to: (1) create filters to select actors from a specific team (as suggested by a study participant), (2) include a timeline in the proposed tool (in order to understanding how agents and activities behave over time), and (3) analyze SPPV's visual scalability.

Acknowledgments

We would like to thank CEOsoftware and Projetus TI, for kindly sharing their data and providing feedback to our research, and CNPq and FAPERJ, for their financial support.

References

Belhajjame, K., BFar, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., Tilmes, C., Moreau, L. and Missier P. (2013) "PROV-DM: The PROV Data Model", W3C.

Chen, P., Plale, B. (2015) "Big Data Provenance Analysis and Visualization", International Symposium on Cluster, Cloud and Grid Computing, pp. 797-800.

Costa, G. C. B. (2016) "Using Data Provenance to Improve Software Process Enactment, Monitoring and Analysis", International Conference on Software Engineering (ICSE 2016), Austin, USA, pp. 875-878.

Dalpra, H. L. O., Costa, G., Sirqueira, T. F. M., Braga, R., Werner, C. M., Campos, F., David, J. M. N. (2015) "Using Ontology and Data Provenance to Improve Software Processes", Proceedings of the Brazilian Seminar on Ontologies, pp. 10-21.

Ellson J., Gansner, E., Koutsofios, L., North, S. C., Woodhull, G. (2001) "Graphviz - open source graph drawing tools", Springer Berlin Heidelberg, pp. 483-484.

Falbo, R., Bertollo, G. (2005) "Establishing a Common Vocabulary for Helping Organizations to Understand Software Processes", Proceedings of the 1st VORTE, Enschede, The Netherlands.

Hoekstra, R., Groth, P. (2014) "PROV-O-Viz-understanding the role of activities in provenance," Provenance and Annotation of Data and Processes", Springer International Publishing, pp. 215-220.

Moreau, L., Groth, P., Cheney, J., Lebo, T., Miles, S. (2015) "The rationale of PROV", Web Semantics: Science, Services and Agents on the World Wide Web.

Murta, L., Braganholo, V., Chirigati, F., Koop, D., Freire, J. (2014) "noWorkflow: Capturing and analyzing provenance of scripts", Proceedings of the Provenance and Annotation of Data and Processes. Springer International Publishing, pp. 71-83.

OMG (2011) "Business Process Model and Notation - Version 2.0", Technical Report.

Schots, M., Werner, C. (2015) "On Mapping Goals and Visualizations: Towards Identifying and Addressing Information Needs", III Workshop de Visualização, Evolução e Manutenção de Software (VEM 2015), Belo Horizonte, Brasil.

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