A Proposal for Sharing Software Process Provenance Data in Heterogeneous Environments*

Gabriella Costa[1], Eldânae Nogueira Teixeira[2], Cláudia Werner[2], Regina Braga[3]

[1]Computer and Mechanics Department
Federal Center for Technological Education of Minas Gerais (CEFET-MG)
36700-001 - Leopoldina - MG - Brazil

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

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

gabriella@cefetmg.br, {danny, werner}@cos.ufrj.br, regina.braga@ufjf.edu.br

Abstract. *Software development practices have evolved, and new approaches have emerged, like Global Software Development (GSD). In addition, software development companies started to adopt data-driven practices in parts of their business. However, using and sharing software process data in a distributed and heterogeneous environment, like the GSD context, could be a challenging topic for many software engineers. In this paper, we present a proposal for sharing software process provenance data using a model that extends PROV, the PROV-SwProcess model. An example of applying this model using a process from the industry that deals with error handling and the implementation of new features in an Enterprise Resource Planning system is presented and explains how the model allows sharing software process provenance data, in addition to providing inferences and insights about these data.*

1. Introduction

In recent years, software development practices have evolved, bringing new ways to develop software [JANSEN, 2020]. New approaches to software development have emerged, such as Global Software Development (GSD) also known as Global Software Engineering (GSE) [HERBSLEB, 2007], and, more recently, Software Ecosystems (SECO) [JANSEN, 2020]. These paradigms aim to reduce software development costs and time through factors such as reusing artifacts developed by third parties, workload distribution, and knowledge sharing, in a distributed scenario. Industry 4.0 solutions consider a complex system of interconnected digital technologies, information systems, and processing technologies that demand high interdependency of competencies and technological complementarity. Then, GSD is inserted in the context of Industry 4.0 and can support its development using various digital technologies such as Cloud Computing, Big Data and Artificial Intelligence.

Software development companies started to adopt data-driven practices in parts of their business over time [JANSEN, 2020]. In this new scenario, companies need to use

* Open peer review artifacts for this paper are available at https://zenodo.org/communities/opensciense2021

data in accounting, marketing, and sales for calculating various performance indicators (such as return on investment for accounting, errors found in deployed products, and defect management). In addition to the use of data-driven practices, software process (SP) is a critical factor for developing quality software products. However, using and sharing SP data in a distributed and heterogeneous environment, could be a challenging topic for many software engineers. Besides that, "sharing data and models is not a simple matter" [MENZIES *et al.*, 2014]. We claim that the use of an adequate data provenance model could support and standardize this activity. In this vein, the main goal of this paper is to *present a proposal for sharing software process provenance data in heterogeneous environments*, such as the GSD context, using a model that extends PROV [GROTH and MOREAU, 2013], the W3C recommended standard.

The remainder of this paper is organized as follows: Section 2 presents the main definitions necessary to understand the approach, as well as works related to the topics involved. Section 3 details PROV-SwProcess, a PROV extension data model for SP, the basis of our proposal. Section 4 describes an example of applying this model using a process from the industry. Finally, Section 5 presents the final remarks and future work.

2. Background

Software process is "a complex endeavor involving professionals, organizations, company policies, tools, and support environments" [LEE *et al.*, 2020]. Software processes are represented by process models. A process model "reflects an organization's know-how regarding software development" including practical experience [MÜNCH *et al.*, 2012] [EISTY *et al.*, 2019]. These models can be prescriptive (describing how something should be done) or descriptive (describing how something is done in reality - process execution data). They can reflect different types of information and different process data sources, ending in heterogeneous data. In this vein, it is important to emphasize that "Software engineering process data is a valuable source of information regarding the history and evolution of a software project" [BACHMANN and BERNSTEIN, 2009] and "Effective management planning, decision-making, and learning processes rely on a spectrum of data, information, and knowledge to be successful" [BASILI *et al.*, 2007]. However, data-driven practices have challenges involving a cycle of activities to prepare the data to be used. This cycle includes strategies for data collection, data storage, data representation, data integration, data sharing, and data maintenance. A first step to appropriately apply this cycle is to understand the possible process elements to be treated, used, and shared. Also, some unified representation of these elements is important to support data collection in GSD, considering multiple sources and maintaining the track to these sources. However, "until now, a single commonly accepted process schema for SP has not been established" [MÜNCH *et al.*, 2012] and "many organizations and projects possess insufficient or poorly organized data collection and analysis mechanisms that result in limited, inaccurate, or untimely feedback to managers and developers" [BASILI *et al.*, 2007].

One possible way to support SP reproducibility, sharing, consensual understanding in a distributed and heterogeneous scenario and reduce the possibility of repeating failed executions is by using provenance. Data provenance can be defined as the description of the origins of a piece of data and the process by which it arrived in a database [BUNEMAN *et al.*, 2001]. Provenance can be divided into two types: (i) prospective provenance that captures a computational task's specification and corresponds to the steps that must be followed to generate a data product, and (ii) retrospective provenance that captures the steps executed as well as information about the environment used to derive a specific data product [FREIRE *et al*., 2008]. Tracking provenance enables sharing, discovering, and reusing data, simplifying collaborative activities in a GSD scenario, in addition to reproducing how something like a build failure was generated, for example [BOSE *et al*., 2019]. Besides that, a provenance model enables "inter-operable interchange of provenance information in heterogeneous environments such as the Web" [GROTH and MOREAU, 2013].

3. A PROV Extension Data Model for Software Processes

Considering that PROV [GROTH and MOREAU, 2013] does not capture the specificities of a SP model, extensions must be done to capture provenance data from SP and also to provide a standard model that helps in the sharing and understanding of SP data in a GSD context. Therefore, the PROV-SwProcess[1] was proposed as a PROV standard extension for SP provenance representation. PROV-SwProcess covers prospective (standard process and intended process) and retrospective provenance (executed process). Besides that, includes the essential aspects of SP: activities, stakeholder, resource, procedure, and artifact, as proposed in SPO [FALBO and BERTOLLO, 2009]. It is divided into (i) associations (or relations), (ii) classes, and (iii) specific inference rules. Figures 1 shows part of the model and the following remarks about the meaning of the used symbols should be considered: (i) constructs and associations presented between "<<>>" were derived from PROV; (ii) elements in yellow ellipses are specializations of the Entity PROV type and elements in orange pentagons are specializations of the Agent PROV type; (iii) associations with black solid lines are used to capture Retrospective Provenance, and associations with red dashed lines can be inferred by PROV-SwProcess and their respective provenance rules, that is, they do not necessarily need to be captured or informed in the SP provenance data.

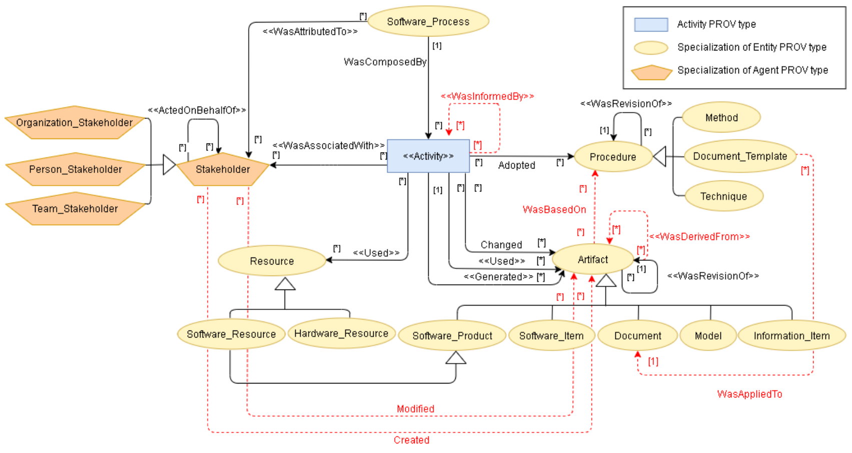

Figure 1. Part of PROV-SwProcess Conceptual Model

4. Sharing Software Process Provenance Data

PROV-SwProcess model works as a standard for sharing information from SP and can be used and understood by globally distributed teams. In addition, the model helps in

[1] The complete ontology of this model version can be found in https://doi.org/10.5281/zenodo.5222104 and its details and reviews are available in http://www.gabriellacastro.com.br/provswprocess/

understanding and tracking changes, all in a uniform way, even considering the heterogeneity of the processes. In order to apply/use the model, the following steps are suggested: (1) Capture/store SP execution data, such as: (i) executed processes with their name and responsible (a Stakeholder); (ii) performed activities of each process, with their name, start, and end time; (iii) Stakeholders associated with the performed activity (mandatory) and their specific role (optional); (iv) Artifacts changed, used, or generated by the performed activity; (v) Procedures adopted for the execution of the performed activity (optional); (vi) Hardware and/or Software resources used by the performed activity (optional); (vii) Responsibility among stakeholders (optional); (viii) Process standard model and process intended model definition, in order to allow process prospective provenance capture and analysis. If the data captured in the first activity are not previously organized according to the PROV-SwProcess model, they must be manipulated and organized/stored according to this model. To make it possible, a generic wrapper should be specialized and configured to make the necessary conversions between different data formats. Therefore, the effort to instantiate the model will depend on how the data coming from the software processes are captured. In this sense, it is necessary to use a wrapper to convert existing data to the structures/constructions proposed by the PROV-SwProcess. Considering that there are several tools for the creation and execution of software processes throughout its lifecycle, the use of a wrapper for this conversion of the stored/captured data to the proposed model is essential. For example, we have wrappers configured for Mantis, for a proprietary version control system, and other that allows converting specific .csv files according to PROV-SwProcess. After storing the SP data in a relational database, modeled according to PROV-SwProcess constructs and relations (e.g., we have tables to store activities, artifacts, stakeholders, wasAssociatedWith relation – which relates activities with the stakeholders who have performed them, etc), an ontology is populated, and an inference machine is executed. Lastly, SPPV (Software Process Provenance Visualization) tool provides two types of visualization (graphs and tables) [COSTA *et al*., 2016] using all the data and new inferred information, helping in understanding and tracking SP provenance data.

As an example of using PROV-SwProcess model to achieve the goal proposed in this paper, a SP from industry that deals with error handling and the implementation of new features in an Enterprise Resource Planning system was chosen. It is from a medium-size company that acts in the software development context for more than thirty years, specifically in creation/maintenance of accounting systems, and all its employees work in home-office and are divided in distributed teams, dealing with different SP. We selected ten executed instances of the SP and they were performed by six different roles (Client, Test Team, Support, Support Manager, Development Manager, and Programmer). Figure 2 shows an example of a generated visualization using SPPV and the provided SP data. It supports in understanding of a shared information about all the stakeholders that act as *Programmer*, *Support* or *Client* in the SP. The group of roles in the lower corner of the figure corresponds to the three roles informed in the process model that had no associated stakeholder (based on the provided execution data). According to this figure, we can also see that the most versatile stakeholder is *Person_1*, who acts as *Programmer* and *Support,* according to the process provenance data. As an insight, in a next instantiation of this process, if the process manager needs to allocate a *Programmer* or a *Support* person in a specific activity, he/she knows who can perform these roles, based on previous execution data. This analysis was performed by one of the provenance model's authors, but she did not participate in the SP. She knew only the basic information about the SP. When this analysis was presented to the SP manager, the following feedback was obtained: (i) the analysis presented before is *correct*; however, it is not common during the process execution that a stakeholder assumes both a *Support* and a *Programmer* role; besides that, he agreed that this occurred in one of the analyzed instances; (ii) this analysis *can* assist in the proposed decision-making; (iii) he *cannot* obtain this analysis using his current process management tool; (iv) the information provided by this visualization is *somewhat relevant* to support in analysis and decision-making processes[2].

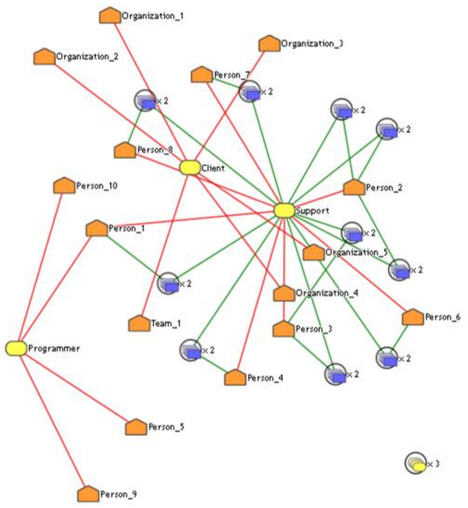

Figure 2. Visualization of SP Provenance Data using SPPV.

5. Final Remarks

In this paper, we present and use a model that can be applied for sharing SP provenance data in heterogeneous environments, such as the GSD context. It works as a standard for sharing information from SP and can be used and understood by globally distributed teams. A SP from the industry is presented and explains that the model allows sharing SP provenance data, in addition to providing inferences and insights about these data. The applicability of the PROV-SwProcess model in sharing SP provenance data can also be validated as presented by Bose *et al*. [2019]. This related work presents BLINKER, an extensible framework that implements/uses the PROV-SwProcess model in a blockchain-based conceptual framework for capturing, storing, exploring, and analyzing software provenance data. On the other hand, the limitation of our proposal as a whole (and not of the model itself) is the need to use a wrapper, as mentioned in Section 4.

As future work, considering the validation and evolution of this proposal, despite presenting an example of using the model in a real scenario, we must apply it to share and analyze the interlacing of data from various projects and processes, considering that it can bring important knowledge and insights about GSD, besides performing a more in-depth empirical evaluation. Considering the PROV-SwProcess model, the following possibilities can be raised: (i) Explore other possibilities of stakeholder relationships (not just acted on behalf of) that could be captured during SP, e.g. collaborative relationships between two or more stakeholders; (ii) check the possibility of deriving new relationships that can be established and/or inferred across the model process levels.

[2] A detailed discussion about all the questions that PROV-SwProcess can answer about the presented example is listed on https://doi.org/10.5281/zenodo.5257245

REFERENCES

BACHMANN, A., BERNSTEIN, A. “Software process data quality and characteristics - a historical view on open and closed source projects”. In: Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, Amsterdam, The Netherlands, August 2009, pp. 119-128, 2009.

BASILI, V., ROMBACH, D., SCHNEIDER, K., KITCHENHAM, B., PFAHL, D., SELBY, R. (Eds.). Empirical Software Engineering Issues. Critical Assessment and Future Directions: International Workshop, Dagstuhl Castle, Germany, June 26-30, 2006, Revised Papers (Vol. 4336). Springer. 2007.

BOSE, R.J.C., PHOKELA, K.K., KAULGUD, V., PODDER, S. Blinker: A blockchain-enabled framework for software provenance. In: 26th Asia-Pacific Software Engineering Conference (APSEC). IEEE, pp. 1-8, December 2019.

BUNEMAN, P., KHANNA, S., TAN, W.C. Why and where: A characterization of data provenance. In: 8th International Conference on Database Theory, London. pp. 4-6, 2001.

COSTA, G. C. B., SCHOTS, M., OLIVEIRA, W. E. B., DALPRA, H. L. O., WERNER, C. M. L., BRAGA, R., DAVID, J. M. N., MIGUEL, M. A., STROELE, V., CAMPOS, F. SPPV: Visualizing Software Process Provenance Data. In: IV Workshop on Software Visualization, Evolution and Maintenance - VII Brazilian Congress on Software: Theory and Practice (CBSoft 2016), pp. 49-56, 2016.

EISTY, Nasir U.; THIRUVATHUKAL, George K.; CARVER, Jeffrey C. Use of software process in research software development: a survey. In: Proceedings of the Evaluation and Assessment on Software Engineering, pp. 276-282, 2019.

FALBO, R. A., BERTOLLO, G. “A software process ontology as a common vocabulary about software processes”. International Journal of Business Process Integration and Management, v. 4, n. 4, pp. 239-250, 2009.

FREIRE, J., KOOP, D., SANTOS, E., SILVA, C. T. Provenance for Computational Tasks: A Survey. Computing in Science and Engineering, vol. 10, no. 3, pp. 11-21, 2008.

GROTH, P., MOREAU, L. “PROV-Overview”, 2013. Available at: <https://www.w3.org/TR/prov-overview/>. Accessed on: jul 2021.

HERBSLEB, J. D. Global software engineering: The future of socio-technical coordination. In: Future of Software Engineering, pp. 188-198. IEEE, 2007.

JANSEN, S., A focus area maturity model for software ecosystem governance, Information and Software Technology, vol. 118, ISSN 0950-5849, 2020.

LEE, J. C., CHEN, C. Y. Exploring the team dynamic learning process in software process tailoring performance: A theoretical perspective. Journal of Enterprise Information Management, vol. 33, no. 3, pp. 502-518, 2020.

MENZIES, T., KOCAGUNELI, E., MINKU L., PETERS, F., TURHAN, B. Sharing data and models in software engineering. Morgan Kaufmann, 2014.

MÜNCH, J., ARMBRUST, O., KOWALCZYK, M., SOTÓ, M. Software process definition and management. Springer Science & Business Media, 2012.