Modelling Knowledge about Software Processes using Provenance Graphs and its Application to Git-based Version Control Systems

Andreas Schreiber
German Aerospace Center (DLR)
Cologne, Germany
andreas.schreiber@dlr.de

Claas de Boer*
German Aerospace Center (DLR)
Berlin, Germany
claas.deboer@dlr.de

ABSTRACT

Using the W3C PROV data model, we present a general provenance model for software development processes and—as an example—specialized models for `git` services, for which we generate provenance graphs. Provenance graphs are knowledge graphs, since they have defined semantics, and can be analyzed with graph algorithms or semantic reasoning to get insights into processes.

KEYWORDS

provenance, software development process, knowledge graphs, git, version control systems

ACM Reference Format:
Andreas Schreiber and Claas de Boer. 2020. Modelling Knowledge about Software Processes using Provenance Graphs and its Application to Git-based Version Control Systems. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3387940.3392220

1 INTRODUCTION

Today, many research fields—including computer science—use *Provenance* [5] to verify data products and to analyse processes that led to them. Provenance can be used to form assessments about quality, reliability or trustworthiness of a piece of data. The knowledge of provenance includes aspects such as sources and processing steps as well as dependencies and contextual information.

Provenance can be expressed in many formats. We focus on the standard W3C PROV [2], which defines the provenance data model PROV-DM [6] to support the interoperable interchange of provenance in heterogeneous environments such as the web. The core structure of PROV-DM relies on the definition of the model class elements *entities* Entity, *activities* Activity, and *agents* Agent that are involved in producing a piece of data or artifact and on definitions of *relations* to relate these class elements, such as *wasGeneratedBy*, *wasAssociatedWith*, *wasAttributedTo*, and *used*. Each of the class elements and relations can have additional attributes.

Recorded or generated provenance of an entity (i.e., a file, a data set, or any other type of artifact) is a *directed acyclic graph* (DAG). Since all nodes and edges of this graph have a defined semantics, the provenance graph is a specific *knowledge graph*. That means for specific uses of PROV-DM, each class element (i.e., entity, activity, and agent) has a least one specialization with a certain semantics.

Our work aims to standardize, generate, and use *provenance of software artifacts* and—closely related to that—*provenance of software development processes* by defining a general PROV model for software developments processes. This includes specializations for all PROV class elements. Our goal is to be able to trace and determine the origin of artifacts such as issues, source code files, build results, or documentation, and to understand and get insights into the process as a whole using requirement specification, developer actions, design decisions, or tools invocations. We distinct between:

- **Prospective provenance** captures how workflows produce artifacts in general (i.e., the "recipe" or possible workflow description), which in our case is covered by a general PROV model for software development (Section 2).
- **Retrospective provenance** is the result of particular executed workflows (e.g., provenance for artifacts that are produced in practice). Our example is provenance for `git` services (Section 3).

2 PROVENANCE OF SOFTWARE ARTIFACTS

Due to the complexity of today's software many development process models evolved, together with many tools. A typical tool suite consists of an integrated development environment (IDE), a version control system, an issue tracker, a continuous integration framework, and a documentation management system.

Based on previous work [10], where we developed a provenance model for software development processes using the *Open Provenance Model* (OPM) notation, we develop an *extensible* PROV model that currently covers issue tracking (requirements, bugs), development (planning, design, coding, testing), continuous integration, documentation (developer, user), and release (Figure 1).

The general model can be extended with further activities such as editing or deployment and further actors such as software bots or software analytics tools. If used for concrete processes, each of the PROV class elements must be defined with specialized class elements. For example, the generic actor role "`User`" has to be specialized to roles such as "`Author`" or "`Test Manager`".

To get knowledge from provenance graphs, one has to query the provenance database. Examples queries include questions related to quality assurance (e.g., "*How many releases have been produced this*

[*] Also with Technische Universität Dresden.

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7963-2/20/05.
https://doi.org/10.1145/3387940.3392220

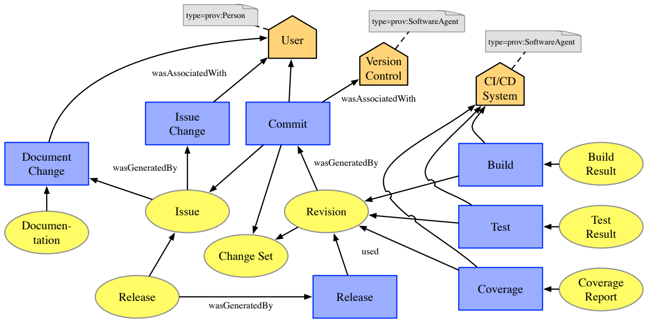

Figure 1: PROV model for software development processes (excerpt; for clarity, some relation types and most attributes are left out).

year?"), process compliance (e.g., "*From which revision was release X built?*"), or developer performance (e.g., "*Which developer is most active in contributing documentation?*").

3 PROVENANCE FOR GIT SERVICES

As a practical example for retrospective provenance, we consider the distributed version-control system `git`, which tracks changes in a file system. Nowadays, `git` is used in many developer workflows. Especially Open-Source projects use `git` via hosting services such as GitHub or GitLab.

Based on the general PROV model (Section 2), we model all actions that are possible with `git` services with more specialized PROV models. Our work relies on the previous works Git2PROV by Nies et al. [7] and GitHub2PROV by Packer et al. [8]. We provide a PROV model for GitLab, GitLab2PROV [1]. For example, for a `commit` it contains the specialized PROV activity `gitlab_commit` and the PROV entities for the modified file `file_v-1` and `file_v` (Figure 2).

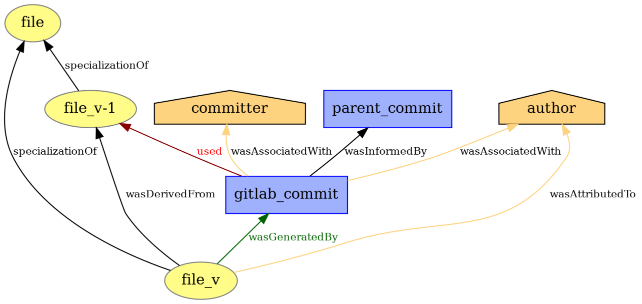

Figure 2: PROV model for commit of a new file version in GitLab.

To practically extract the PROV graph from repository's, the respective implementations use the command "`git log`" (Git2PROV) or the API's (GitHub2PROV, GitLab2PROV). GitLab2PROV stores the PROV graph either in a text file (*.provn* format)[1] or in a graph database (Neo4j).

Performing queries, graph reasoning, or extracting knowledge otherwise is possible by using Cypher queries (Listing 1) or graph algorithms in Neo4j. For knowledge, we consider the semantics and an ontology for `git` [4].

Listing 1: Cypher query for the number of files that were edited by an agent.

```
MATCH
  (user:Agent)-[:wasAttributedTo]-(fileVersion:Entity),
  (fileVersion:Entity)-[:specializationOf]->(file:Entity)
WHERE
  fileVersion.prov:type = "file_version" AND
  file.prov:type = "file"
RETURN
  user.name, COUNT(DISTINCT file) AS file_count
ORDER BY file_count DESC
```

4 CONCLUSIONS AND FUTURE WORK

We presented a draft PROV model for software development processes and an example for retrospective provenance. The resulting provenance graphs are knowledge graphs given their specified semantics for nodes and edges.

Future work includes to add more sources to the provenance graph. For example, from design documents (UML2PROV [9]) or IDE's. The provenance graph can also be extended with provenance for (running) algorithms, data, or machine learning processes [3].

REFERENCES

[1] Claas de Boer and Andreas Schreiber. 2020. *DLR-SC/gitlab2prov: Initial release.* https://doi.org/10.5281/zenodo.3624167

[2] Paul Groth and Luc Moreau. 2013. PROV-overview. An overview of the PROV family of documents. (2013).

[3] Sophie F. Jentzsch and Nico Hochgeschwender. 2019. Don't Forget Your Roots! Using Provenance Data for Transparent and Explainable Development of Machine Learning Models. In *1st International Workshop on Explainable Software (EXPLAIN 2019)*.

[4] Dennis Oliver Kubitza, Matthias Böckmann, and Damien Graux. 2019. Towards Semantically Structuring GitHub. In *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019.* 141–144.

[5] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, and Laszlo Varga. 2008. The provenance of electronic data. *Commun. ACM* 51, 4 (2008), 52–58. https://doi.org/10.1145/1330311.1330323

[6] Luc Moreau, Paolo Missier, Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. 2013. PROV-DM: The PROV Data Model. http://www.w3.org/TR/2013/REC-prov-dm-20130430/

[7] Tom De Nies, Sara Magliacane, Ruben Verborgh, Sam Coppens, Paul T. Groth, Erik Mannens, and Rik Van de Walle. 2013. Git2PROV: Exposing Version Control System Content as W3C PROV. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013.* 125–128.

[8] Heather S. Packer, Adriane Chapman, and Leslie Carr. 2019. GitHub2PROV: Provenance for Supporting Software Project Management. In *11th International Workshop on Theory and Practice of Provenance (TaPP 2019)*. USENIX Association, Philadelphia, PA. https://www.usenix.org/conference/tapp2019/presentation/packer

[9] Carlos Sáenz-Adán, Luc Moreau, Beatriz Pérez, Simon Miles, and Francisco José García Izquierdo. 2018. Automating Provenance Capture in Software Engineering with UML2PROV. In *Provenance and Annotation of Data and Processes - 7th International Provenance and Annotation Workshop, IPAW 2018, London, UK, July 9-10, 2018, Proceedings.* 58–70.

[10] Heinrich Wendel, Markus Kunde, and Andreas Schreiber. 2010. Provenance of Software Development Processes. In *Provenance and Annotation of Data and Processes - Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, June 15-16, 2010. Revised Selected Papers.* 59–63.

[1] As an example, see the PROV file for the GitLab project https://gitlab.com/gitlab-org/gitlab-runner-docker-cleanup at https://openprovenance.org/store/documents/1990.