An Interactive Dashboard for Visualizing the Provenance of Software Development Processes

Andreas Schreiber, Lynn von Kurnatowski, Annika Meinecke, and Claas de Boer
Institute for Software Technology
German Aerospace Center (DLR)
Köln, Weßling & Dresden, Germany
Email: {andreas.schreiber,lynn.kurnatowski,annika.meinecke,claas.deboer}@dlr.de

***Abstract*—Software development is a complex process involving many people and development tools and their interactions; during development, a lot of data such as source code, documents, or software artifacts and information such as issues, discussions, or code analyses are generated or modified. In addition to the analysis and visualization of software systems, it is useful to analyze the software development process to obtain better information about the quality, reliability, and trustworthiness of the software. To gain insights and knowledge about software development processes, we extract the provenance of development processes, especially from version control systems for Git-based software projects, and visualize the provenance information using graph visualization, metrics representation, and development timelines; including an integration of these methods into a web-based dashboard. With the help of visual provenance representations, project managers can gain an overview and insights into development progress, effects of process changes, and interactions among developers and with external contributors, which we exemplify with a case study of a software with high social relevance.**

***Index Terms*—provenance, software visualization, software development processes, visual analytics, interactive dashboards**

I. Introduction

Software is an important innovation factor and a crucial part in modern research and development areas. However, the process of software development is complex and is becoming increasingly complex. The individual process steps are executed by people who interact with tools or with each other. To understand the software development process and thus make better statements about the quality, reliability and trustworthiness of the resulting software product, one can record and analyze the *provenance* [1] of the process or the resulting artifacts.

Provenance can be expressed in many ways. We use the W3C specification PROV [2], which defines the provenance data model PROV-DM [3] and an ontology PROV-O [4], among others. PROV was inspired by various different approaches [5], that is adaptable to any domain.

One way to analyze provenance information and thereby gain insight into the process under investigation is through visual analytics using methods such as graph visualization or visual representation of provenance metrics. We contribute with *visual analytics methods for retrospective provenance of software development processes*, which we extract by repository mining from code hosting environments based on the version control system *git*.

We evaluate our visualizations using the *luca App*[1] as a case study. The luca App is a mobile app for providing data for contact tracing and risk contact notification during a pandemic. In Germany, there is a controversial discussion about the app, which currently makes it a socially relevant software application. Many IT security experts criticize the app for its security gaps and data protection shortcomings [6]. Nevertheless, many German states have bought it and made it mandatory for checking in at restaurants and stores. For this reason, the app and its development process are interesting objects of investigation.

Most of the development of the luca App is done publicly on GitLab[2] in eight repositories. Development began in February 2021, with 13 people committing changes to git and 42 people contributing changes to issues as of early June[3].

The necessary background information and our contributions are structured as follows:

- We describe our method and tools for retrospectively extracting provenance from the code hosting platforms GitHub and GitLab (Section II).
- We show our visualizations of software development process provenance, in particular graph visualization, visualization of different metrics, and summarizing them in a web-based dashboard (Section III).
- Finally, we summarize related work (Section IV).

II. Provenance of Software Development Processes

For software development processes based on `git` repositories, we extract *retrospective provenance* [7] with data mining on the repositories; in the case of GitHub and GitLab, this also includes provenance information from the respective issue trackers and release systems. The resulting provenance data then contains all modeled activities (e.g., commits, issues changes, releases), the generated or changed entities (e.g., source code files or issues), and involved agents (e.g., developers, testers, or users) along with their relations.

[1] https://www.luca-app.de
[2] https://gitlab.com/lucaapp
[3] https://cauldron.io/project/4448

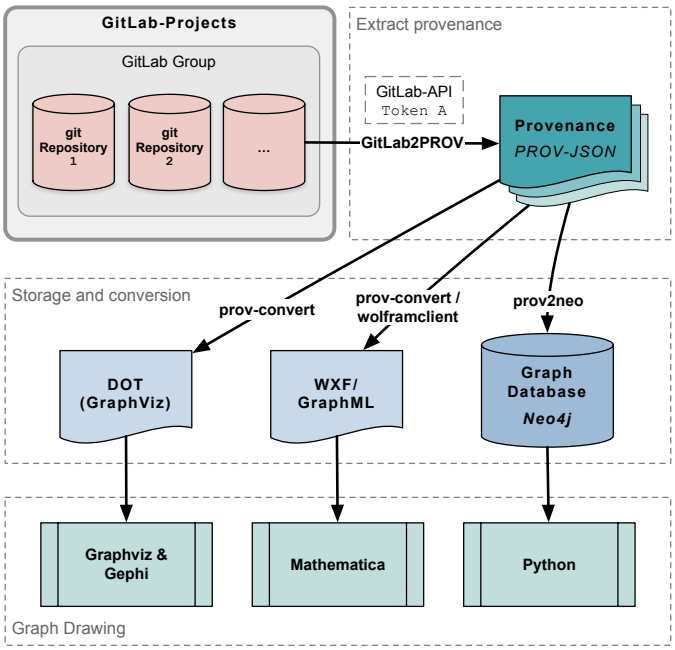

Fig. 1. Extracting provenance from projects hosted on GitLab (i.e., git repositories, issues, releases, etc.) to PROV documents (i.e., files in PROV-JSON format), storing in the graph database Neo4j, converting to graph file formats for visualization with tools such as Gephi, Graphviz, or Mathematica.

For `git`-only repositories, GIT2PROV [8] extracts provenance using the `git` command line tool. GITHUB2PROV [9] extends GIT2PROV and additionally extracts information from the GitHub Issue Tracker using the GitHub API. Our tool GITLAB2PROV [10] uses only the GitLab API. All three tools generate the provenance information in the form of the text representation PROV-JSON, which is converted to other formats for further analysis and visualization and imported as a property graph into the graph database Neo4j (Figure 1). In particular, for graph visualization we use the three tools GEPHI [11], MATHEMATICA and Python with the libraries NETWORKX and PLOTLY; in addition to the graph visualization function of Neo4j.

III. VISUALIZATION OF SOFTWARE ENGINEERING PROVENANCE USING AN INTERACTIVE DASHBAORD

The information from the provenance graph can be visually represented in very different ways. The visualizations are derived from the questions that the developers, project managers or other stakeholders have to the development process. The goal is always to gain insights into the development process that are not directly provided by the development tools. It is therefore a matter of extracting and presenting the knowledge contained in the provenance graph about the activities performed by the developers, the files and issues created or changed, or the responsibilities of the developers using suitable queries.

The queries and visualizations can be related to only a single project or a group of projects—depending on the question. This can be scaled up, for example, to perform repository mining on many projects (for example, all projects in an organization or all projects with certain characteristics) to identify differences or common patterns.

Currently, our provenance dataset only contains the provenance information extracted with GITLAB2PROV. In particular, this means that there is no information about the semantics and content of the individual entities (i.e., the source code files and the texts in issues and commit messages). Also no information about modules, packages, or the architecture of the software system is included.

Depending on the question and the intended visual representation, we define appropriate CYPHER[4] queries on the provenance information stored in the Neo4j database. The principle is always similar: First, we define whether information about the PROV class elements Entities, Activities or Agents is required. Then we further limit the query by specifying relations (i.e., in which structure the PROV class elements are related) and by limiting the query to certain attributes (for example time periods, types, or names).

To visually represent the provenance information (i.e., the results of the CYPHER queries) we use several visualizations, such as graph visualizations, metrics visualizations (e.g., bar charts), time-oriented visualizations (e.g., Sankey charts), task-oriented and work process-oriented visualizations (e.g., Gantt charts), or hierarchy-oriented visualizations (e.g., treemap charts).

A. Interactive Dashboard

To provide more interactivity and to be able to provide the visualizations in the form of a web application, we can compile the individual visualizations into a web-based *interactive dashboard* (Figure 2).

The dashboard contains a selection of the possible visualizations, which are integrated and equipped with interactive controls (Figure 3).

In the "PROV Metrics" section, one can select the projects whose metrics are to be displayed. In the "Event Timeline" the selection of exactly one project is possible. The "Entities Timeline" refers to all projects, since it concerns here the development activities over the time; here one can select the resampling frequency, to represent for example daily, weekly, or quarterly activity progressions. In the "Agent-Entity Relations" area, the individual developers and the files they edited are displayed on the left; if you select one of the developers, the treemap on the right displays the changed files in a hierarchical representation.

To implement the visualizations and dashboard, we use Python with the libraries PLOTLY, NETWORKX, and DASH; in addition to the basic libraries PY2NEO to query the Neo4j database and PANDAS to store the query results and prepare the data.

Currently, the layout and integrated visualizations are defined in Python code; likewise, the connected Neo4j database is fixed.

[4] https://neo4j.com/developer/cypher/

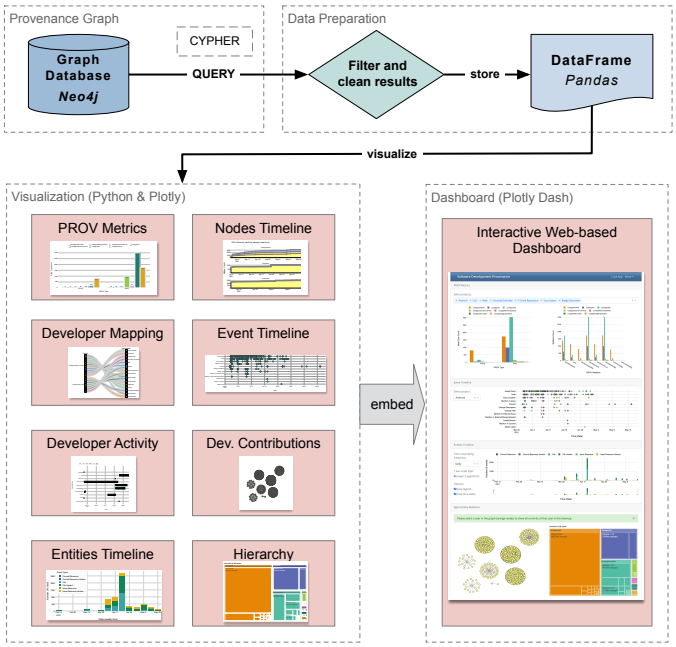

Fig. 2. Provenance visualization flow: querying the Neo4j database with CYPHER queries, storing query results and filtering and data cleansing with Pandas, visualizing with Python and PLOTLY, and displaying in the web-based dashboard with PLOTLY DASH.

B. Insights

To gain a basic understanding of the project and its development history, we considered the metrics below, which led to the following insights:

a) How many activities have been conducted and how many files have been produced or changed?: The provenance graph contains a total number of 6.790 nodes, divided into the different PROV types *activities* and *entities*. The "PROV Type" chart ("PROV Metrics"; Figure 3) shows that the provenance graph of each project contains more entities than activities. This indicates that in all three projects more artifacts and artifacts versions are created than user interactions are performed.

b) What and how many interactions took place for each of the `git` projects?: Similar to the number of nodes, the number of edges of the different PROV relations, provides information about the activity of a project. As shown in the "PROV Relations" chart ("PROV Metrics"; Figure 3), there are only few *wasInvalidatedBy* relations in all three repositories. Relations of this type are used for the removing of a file by a commit. As a result, it can be concluded that few files used for removal in a commit. However, all three luca App repositories contain many *wasAttributedTo* and *wasGeneratedBy* relations. This directly correlates with the number of entities and activities. This is because each entity is generated by an activity, so there is a *wasGeneratedBy* relation for each entity. In contrast to the other two repositories, it is noticeable that particularly in the repository `lucaapp/android` the used relations occurs frequently. The *wasGeneratedBy* relation is used in connection with the reuse of artifacts when generating new artifact versions.

c) What are developers interactions over time?: We examined the interactions of developers over time. The exploration in the "Event Timeline" section (Figure 3) shows that developers rewarded many award emojis to the project in the first month after the initial commits. In addition, many notes were created in the beginning, and new issues were created. This indicates that especially when the project was first made accessible through GitLab, many developers were excited about the project, rewarding it with emoji awards. In addition, developers took part in the discussions surrounding the project leading. Later commits gain no or only few award emojis, which could be an indication of minor commits or decreasing excitement for the project.

d) What is the distribution of events over time?: The "Entities Timeline" (Figure 3) shows the distribution of GitLab events over time for all `lucaapp` repositories, with events being summarized by weeks. Events are further aggregated to events creating new commits, files and issues (Commit Ressource, File, Issue Ressource) and events changing existing commits, files and issues (Commit Ressource Version, File Version, Issue Ressource Version). With only few events in February and March of 2021, the most activity takes place in the first three weeks of April. In the first week, the most amount of issues are generated. By far the most files are created, as well as changed in the third week of April. After these three weeks the activity decreases again, with mostly file version events occurring during the next weeks.

e) How many commits were made by developers?: We illustrated the amount of commits made by users to the different repositories using a treemap visualization. The different repositories are visually separated by color, the amount of changes per commit defines the size of the individual tiles ("Agent-Entity Relations"; Figure 3). Most changes occurred during the commits of `lucaapp/web` (3077 file changes in total), especially during the initial commit, followed by `lucaapp/android` (1093 file changes) and `lucaapp/ios` (1088 file changes). In addition, the treemap can be filtered by individual users, giving the option to display the amount of commits and changes done by a single developer.

IV. RELATED WORK

There are numerous approaches to *visualizing provenance information*; many of them are based on graph visualization and originate from applications where provenance of scientific workflows is studied: *Prov Viewer* [12] is a graph-based visualization tool for visual exploration of PROV graphs, *Provenance Map Orbiter* [13] focuses on interactive exploration of large provenance graphs, *AVOCADO* [14] visualizes provenance based on data flows in biomedical research, and *PROV-O-Viz* [15] is a web-based PROV visualization tool that leverages Sankey diagrams and adds a number of provenance specific features.

Fig. 3. Web-based interactive dashboard. A selection of possible visualizations are integrated and equipped with interactive controls.

In *software visualization and repository mining*, many visualization approaches focus on metrics and visualizations of the source code, for example by Nuzrath et al. [16] or by Pinzger et al. [17]. Very similar to our work is the work of Grabner et al. [18] who extract data from version management, issue tracker, and CI system, store it in a database, and present it in web-based interactive visualizations. Also very similar are the works of Curt et al. [19], who collect software trails from version control and issue tracking systems to build a provenance graph that helps understand software releases.

V. Conclusion and Future Work

We have described how we visualize provenance information from `git`-based software development projects in an interactive dashboard; specifically, projects hosted with GitLab. To do this, we extract provenance using the GitLab2PROV tool, store the provenance graph in the Neo4j graph database, and query for relevant information using Cypher to generate visualizations for questions of interest. We show the visualizations using the socially relevant *luca App* as a case study.

In contrast to other related work for analysis and visualization of development processes, we use the *standardized data model* PROV-DM. With it we can connect the provenance information with those of other applications (e.g., provenance of the execution of the developed software or of the data generated with the software).

Our future work will focus on these areas:

- Further development of visualizations for individual aspects of software development provenance.
- Further development of the dashboard: adapting the visual design and style by developing a consistent visual concept (including users studies) and technical enhancements, such as a search function, the possibility to start GitLab2PROV directly with a repository URL or improvements to higher scalability.

Availability

GitLab2PROV is available as Open Source software under the MIT license: https://github.com/DLR-SC/gitlab2prov

The interactive dashboard with data for the *luca App* repositories is available at https://prov-dashboards.net.

References

[1] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga, "The provenance of electronic data," *Communications of the ACM*, vol. 51, no. 4, pp. 52–58, 2008.

[2] L. Moreau and P. T. Groth, *Provenance: An Introduction to PROV*, ser. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2013.

[3] L. Moreau, P. Missier, K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, and C. Tilmes, "PROV-DM: The PROV data model," 30 April 2013 2013. [Online]. Available: http://www.w3.org/TR/2013/REC-prov-dm-20130430/

[4] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao, "PROV-O: The PROV ontology," 30 April 2013 2013. [Online]. Available: http://www.w3.org/TR/2013/REC-prov-o-20130430/

[5] L. Moreau, P. Groth, J. Cheney, T. Lebo, and S. Miles, "The rationale of PROV," *Web Semant.*, vol. 35, no. P4, p. 235–257, Dec. 2015. [Online]. Available: https://doi.org/10.1016/j.websem.2015.04.001

[6] T. Stadler, W. Lueks, K. Kohls, and C. Troncoso, "Preliminary analysis of potential harms in the Luca tracing system," Mar. 2021.

[7] T. McPhillips, S. Bowers, K. Belhajjame, and B. Ludäscher, "Retrospective provenance without a runtime provenance recorder," in *Proceedings of the 7th USENIX Conference on Theory and Practice of Provenance*, ser. TaPP'15. USA: USENIX Association, 2015.

[8] T. De Nies, S. Magliacane, R. Verborgh, S. Coppens, P. Groth, E. Mannens, and R. Van De Walle, "Git2PROV: Exposing version control system content as W3C PROV," in *Proceedings of the 12th International Semantic Web Conference (Posters and Demonstrations Track) – Volume 1035*, ser. ISWC-PD '13. CEUR-WS.org, 2013, pp. 125–128.

[9] H. S. Packer, A. Chapman, and L. Carr, "GitHub2PROV: Provenance for supporting software project management," in *Proceedings of the 11th USENIX Conference on Theory and Practice of Provenance*, ser. TAPP'19. USA: USENIX Association, 2019.

[10] A. Schreiber, C. de Boer, and L. von Kurnatowski, "GitLab2PROV—provenance of software projects hosted on gitlab," in *13th International Workshop on Theory and Practice of Provenance (TaPP 2021)*. USENIX Association, Jul. 2021. [Online]. Available: https://www.usenix.org/conference/tapp2021/presentation/schreiber

[11] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154

[12] T. Kohwalter, T. Oliveira, J. Freire, E. Clua, and L. Murta, "Prov viewer: A graph-based visualization tool for interactive exploration of provenance data," in *Provenance and Annotation of Data and Processes*, M. Mattoso and B. Glavic, Eds. Cham: Springer International Publishing, 2016, pp. 71–82.

[13] P. Macko and M. Seltzer, "Provenance map orbiter: Interactive exploration of large provenance graphs," in *3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP 11)*. Heraklion, Crete Greece: USENIX Association, Jun. 2011. [Online]. Available: https://www.usenix.org/conference/tapp11/provenance-map-orbiter-interactive-exploration-large-provenance-graphs

[14] H. Stitz, S. Luger, M. Streit, and N. Gehlenborg, "AVOCADO: Visualization of workflow–derived data provenance for reproducible biomedical research," *Computer Graphics Forum*, vol. 35, no. 3, pp. 481–490, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12924

[15] R. Hoekstra and P. Groth, "PROV-O-Viz – understanding the role of activities in provenance," in *Provenance and Annotation of Data and Processes*, B. Ludäscher and B. Plale, Eds. Cham: Springer International Publishing, 2015, pp. 215–220.

[16] S. Nuzrath, N. H. Amarasinghe, K. T. Liyanage, K. Suriyawansa, D. P. Madanayake, and N. Kodagoda, "gCodex: A tool to analyze software repositories over time (visualization)," in *2019 International Conference on Advancements in Computing (ICAC)*, 2019, pp. 174–179.

[17] M. Pinzger, H. Gall, M. Fischer, and M. Lanza, "Visualizing multiple evolution metrics," in *Proceedings of the 2005 ACM Symposium on Software Visualization*, ser. SoftVis '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 67–75. [Online]. Available: https://doi.org/10.1145/1056018.1056027

[18] J. Grabner, R. Decker, T. Artner, M. Bernhart, and T. Grechenig, "Combining and visualizing time-oriented data from the software engineering toolset," in *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2018, pp. 76–86. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/VISSOFT.2018.00016

[19] F. Curty, T. C. Kohwalter, V. Braganholo, and L. Murta, "An infrastructure for software release analysis through provenance graphs," *CoRR*, vol. abs/1809.10265, 2018. [Online]. Available: http://arxiv.org/abs/1809.10265