An overview of Provenance and it's use cases

Carina Haupt

German Aerospace Center (DLR)
Institute for Software Technology

@caha42

Reproducibility > Metadata > Provenance

Reproducibility in (data) science is based on a lot of things (OSS, RDM, ELNs, open formats, etc.), **metadata is one of them**.

Metadata can include a large variety of information (quality, geographic, instrument configuration, file size, etc.), **provenance is one of them**.

Provenance

- Provenance refers to the **source of information** and the **process that led to its existence**
- Provenance information is critical to users trying to understand where a particular data file came from
- Important for audits and assumptions: **certification**, **compliance**, **trust**

Other and related terms: Traceability, Lineage, Logging, Monitoring

More Formal Definition of Provenance

Provenance is

information about entities, activities, and people

involved in

producing a piece of data or thing,

which can be used to form

assessments about its quality, reliability or trustworthiness.

→ *W3C Specification „PROV“*

PROV W3C Working Group
https://www.w3.org/TR/prov-overview

PROV Elements

Entities

- Physical, digital, conceptual, or other kinds of things (artifacts)
- For example, documents, web sites, graphics, or data sets

Activities

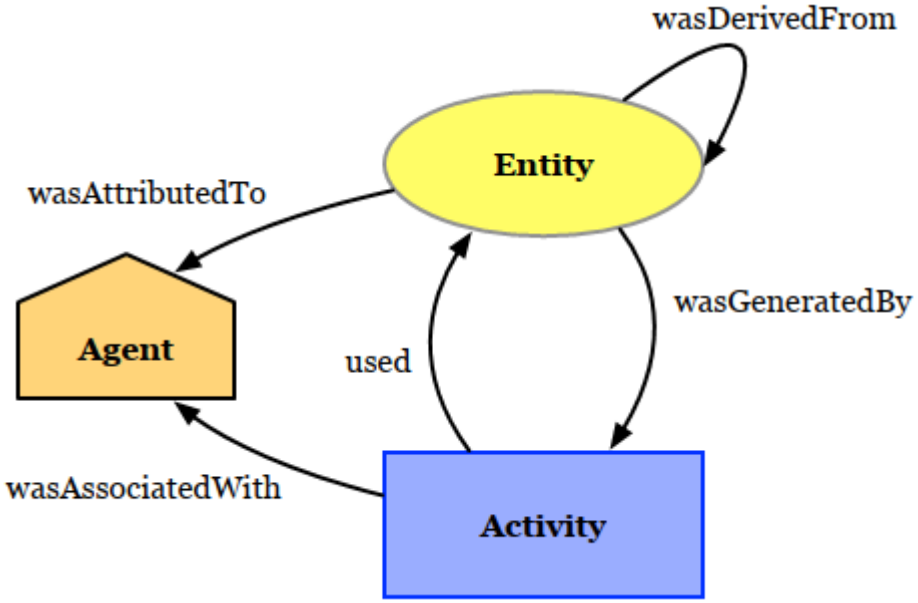
- Activities *generate* new entities or make *use* of existing entities
- Activities could be actions or processes

Agents

- Agents takes a role in an activity and have the responsibility for the activity
- For example, persons, pieces of software, or organizations

Relations between those elements

- Generation, usage, communication, derivation, etc.



Provenance is a *Directed Acyclic Graph (DAG)*

Around PROV

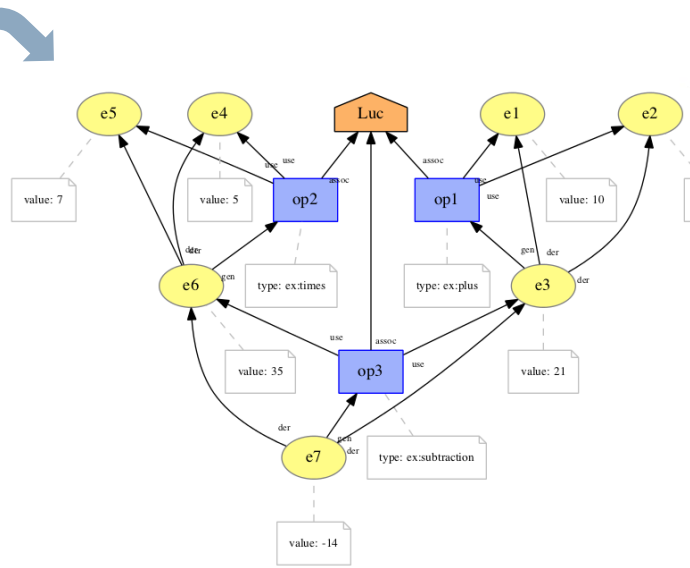
Textual Representations (*PROV-N, JSON, Turtle, XML, …*)

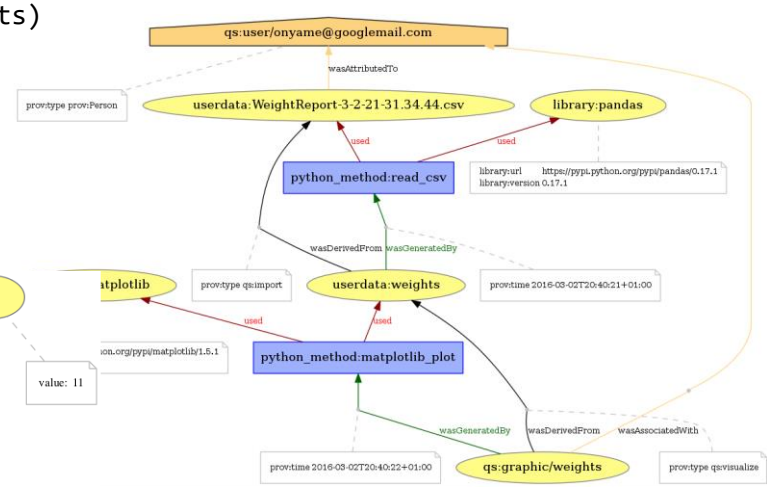
Visualizations

Templates

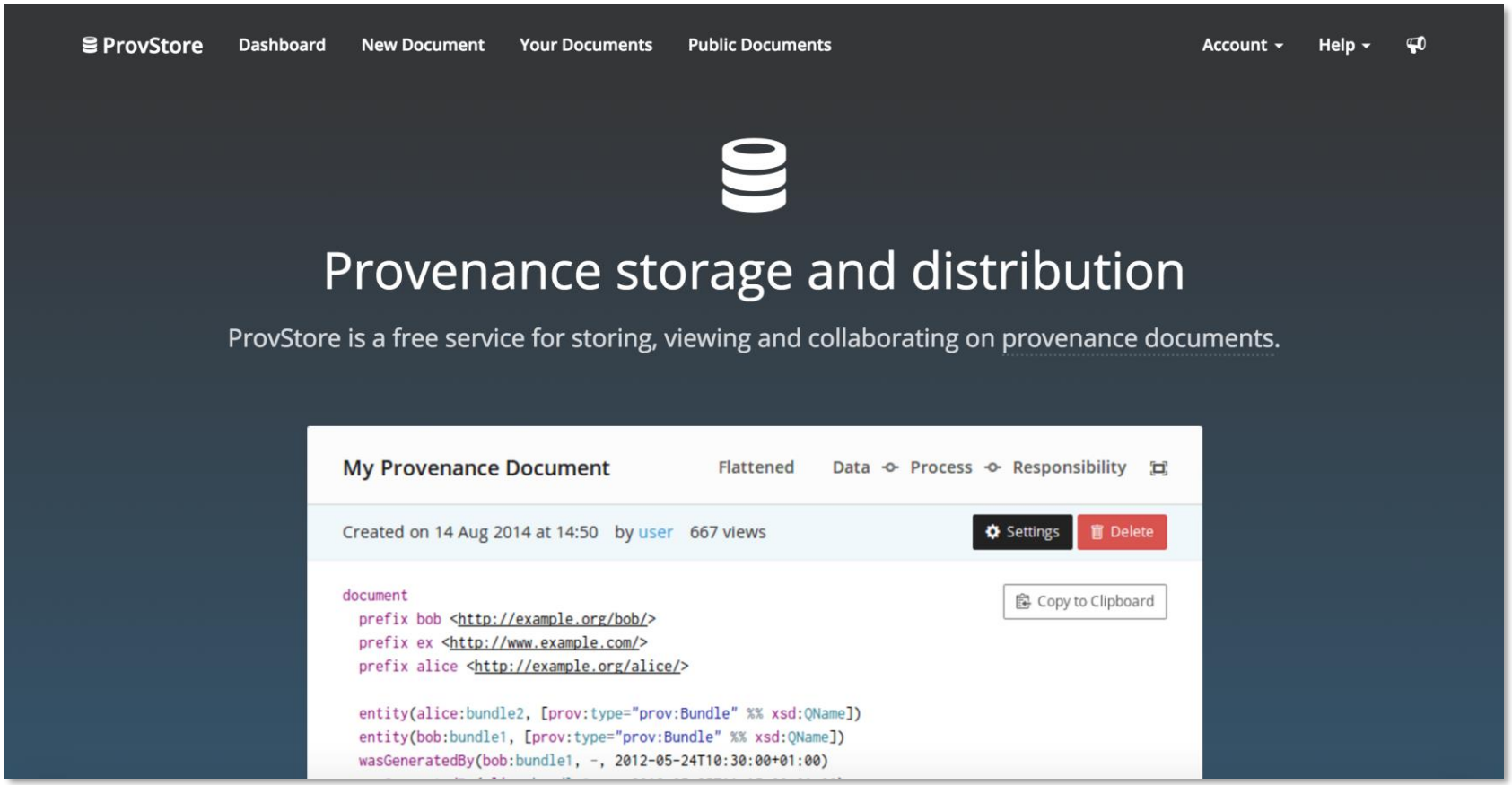
```
document
 prefix tmpl <http://openprovenance.org/tmpl#>
 prefix var <http://openprovenance.org/var#>
 prefix vargen <http://openprovenance.org/vargen#>

 bundle vargen:b
  activity(var:operation, [ prov:type='var:operation_type' ] )
  agent(var:agent)
  wasAssociatedWith(var:operation,var:agent,-)
  entity(var:consumed1,[prov:value='var:consumed_value1'])
  entity(var:consumed2,[prov:value='var:consumed_value2'])
  used(var:operation, var:consumed1, - )
  used(var:operation, var:consumed2, - )
  entity(var:produced,[prov:type='var:produced_type',
     prov:value='var:produced_value'])
  wasGeneratedBy(var:produced, var:operation, - )
  wasDerivedFrom(var:produced, var:consumed1)
  wasDerivedFrom(var:produced, var:consumed2)
 endBundle
endDocument
```

```
document
  prefix userdata http://software.dlr.de/qs/userdata/
. . .
  wasDerivedFrom(userdata:weights, userdata:WeightReport.csv,
  wasDerivedFrom(qs:graphic/weights, userdata:weights,
  wasAssociatedWith(qs:graphic/weights, qs:user/onyame@gmail.com, -)
  used(python_method:read_csv, library:pandas, -)
  used(python_method:matplotlib_plot, userdata:weights, -)
  used(python_method:matplotlib_plot, library:matplotlib, -)
  used(python_method:read_csv, userdata:WeightReport.csv, -)
  wasAttributedTo(userdata:WeightReport.csv, qs:user/onyame@gmail.com)
  agent(qs:user/onyame@gmail.com, [prov:type="prov:Person"])
  entity(library:pandas, [library:version="0.17.1"])
  entity(userdata:WeightReport.csv)
  entity(userdata:weights)
. . .
endDocument
```





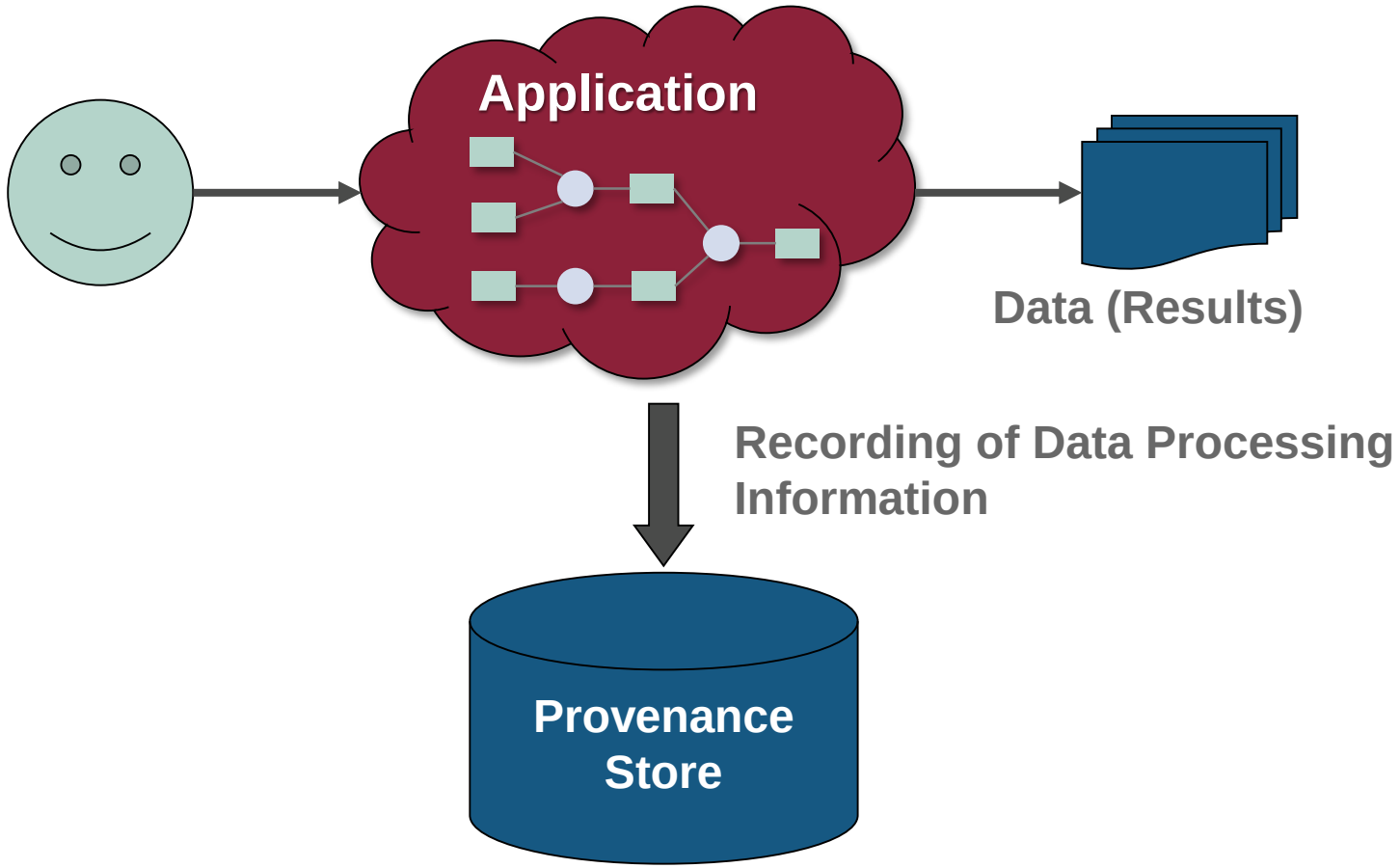
ProvStore (*King's College London*)



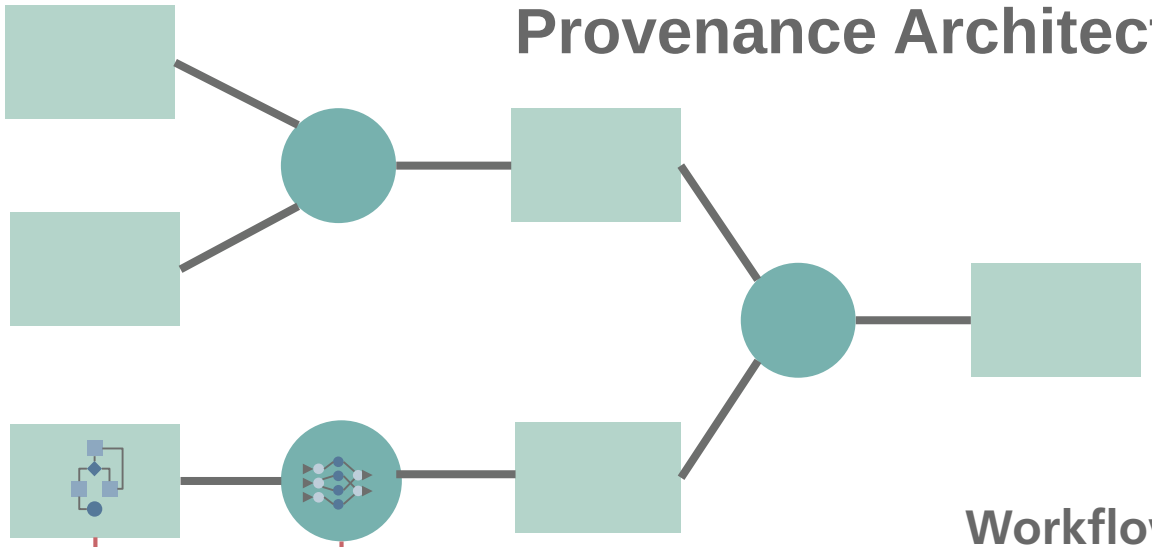
- RESTful web service
- storage and access of provenance documents
- Public and private documents
- Conversion to various text formats
- Simple visualizations
- APIs
  - Python
  - jQuery

https://openprovenance.org/store/

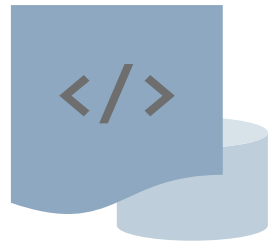
Provenance Architecture



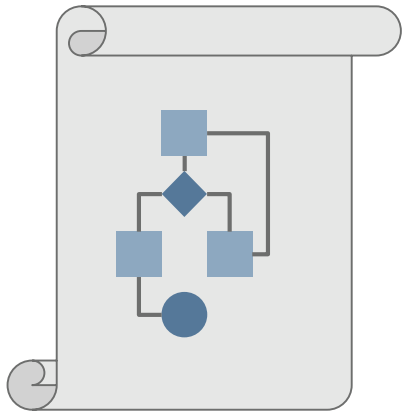
Provenance Architecture at DLR



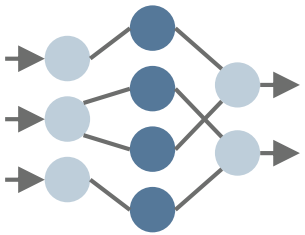
Workflows



Software Development

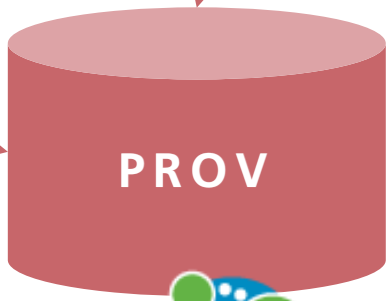


Algorithms / Scripts



Machine Learning

Data Management



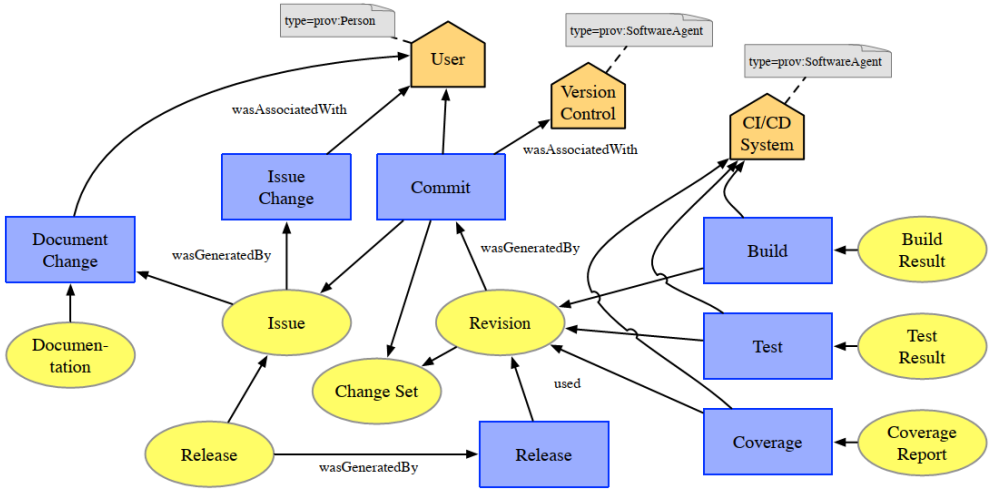
Provenance Store

neo4j

Provenance of Software Artifacts and Development Processes

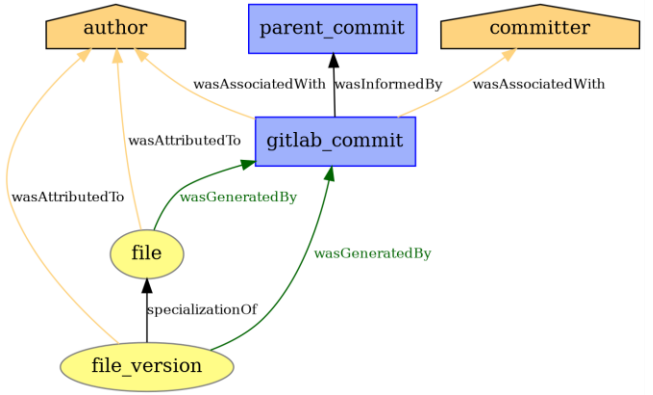
Prospective provenance

- Captures how workflows produce artifacts in general
- For example, the intended **development process**
- Our case is covered by a **general PROV model for software development**



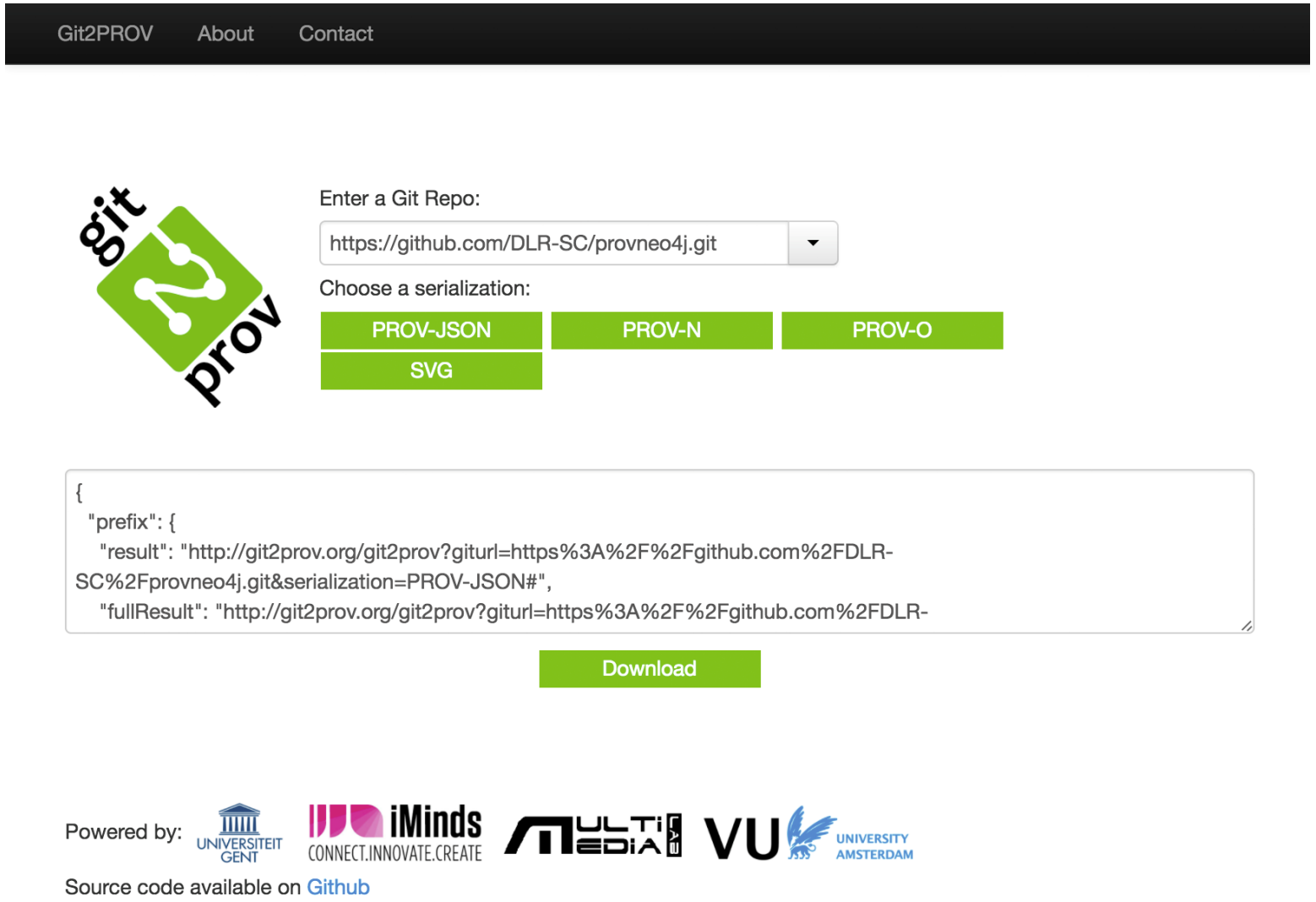
Retrospective provenance

- The result of particular executed workflows (i.e., provenance for artifacts that are produced in practice)
- For example, **provenance of software artifacts** such as source code, build results, or documentation
- Our example is **provenance for git services**



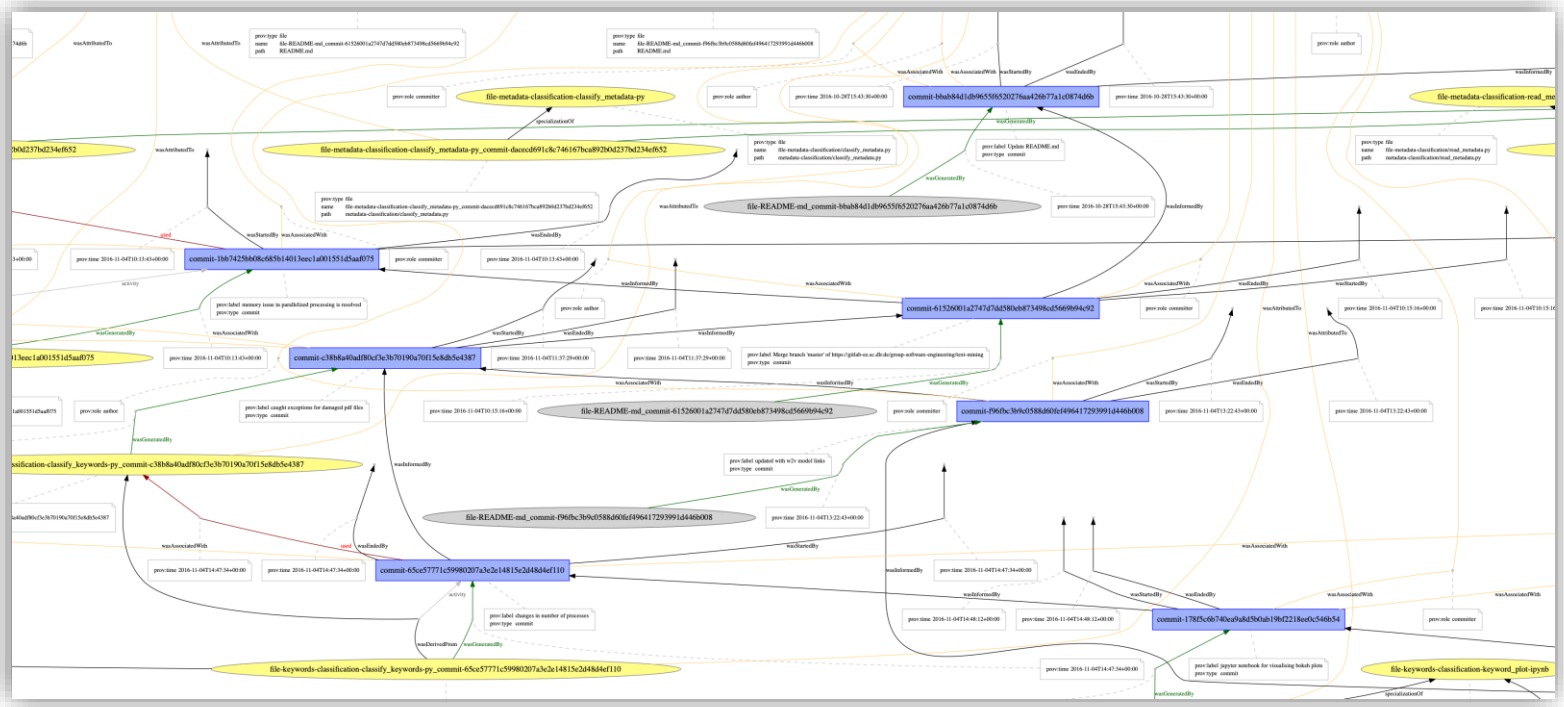
GIT2PROV
http://git2prov.org

- Generate PROV documents from git repositories



Gitlab2PROV
https://github.com/DLR-SC/gitlab2prov

- Generate PROV documents from GitLab projects
  - Files (commits)
  - Issues
  - Comments
- Uses the GitLab-API



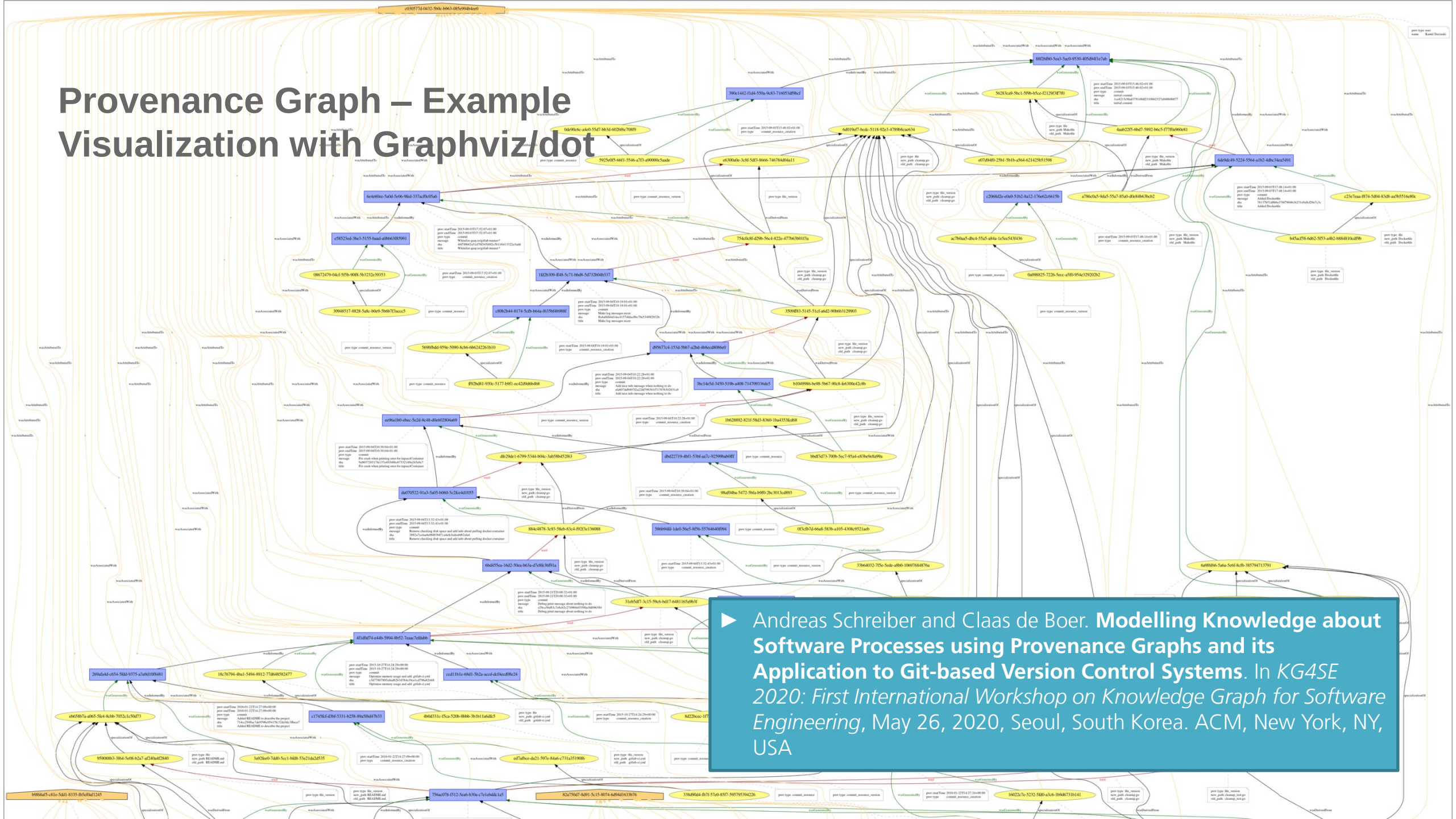
https://openprovenance.org/store/documents/1975

PROVNEO4J – Storing PROV Documents in Neo4j
https://github.com/DLR-SC/provneo4j

```
import provneo4j.api

provneo4j_api = provneo4j.api.Api(
        base_url="http://localhost:7474/db/data",
        username="neo4j", password="python")

provneo4j_api.document.create(prov_doc, name="MyProv")
```

New version PROV2NEO coming soon!
https://github.com/DLR-SC/prov2neo

Provenance Graph – Example Visualization with Graphviz/dot

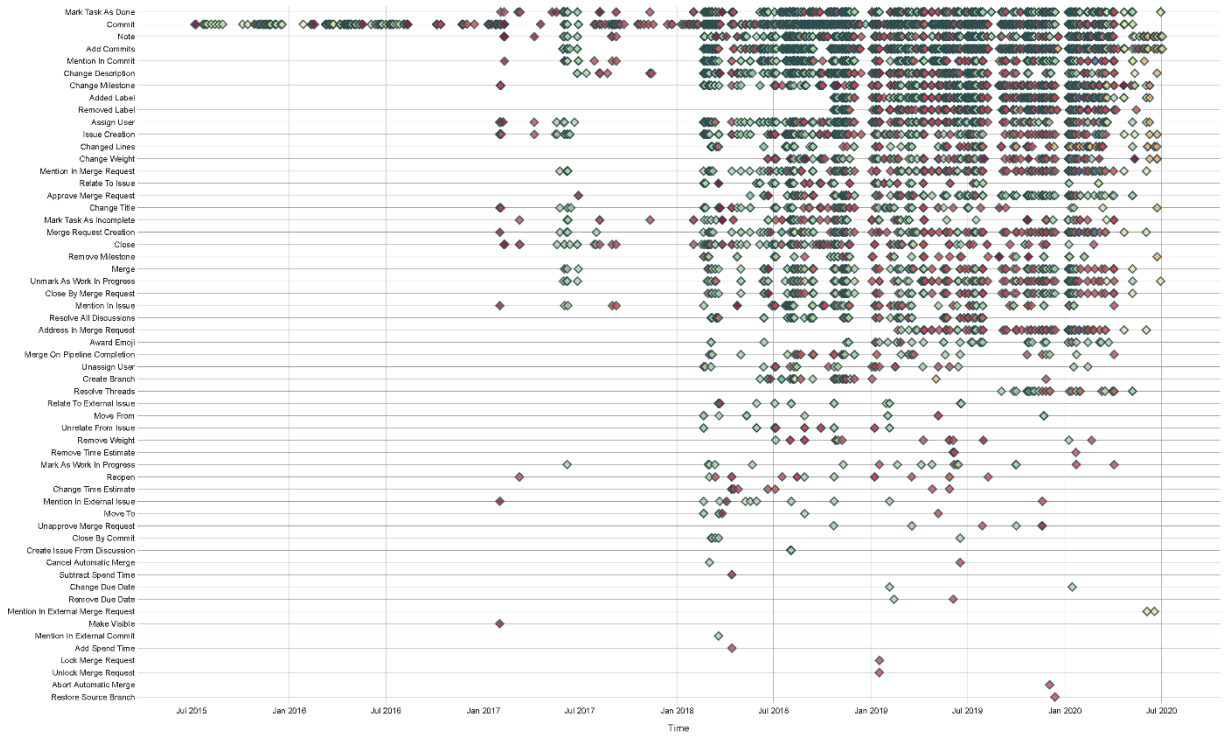


- Andreas Schreiber and Claas de Boer. **Modelling Knowledge about Software Processes using Provenance Graphs and its Application to Git-based Version Control Systems**. In *KG4SE 2020: First International Workshop on Knowledge Graph for Software Engineering*, May 26, 2020, Seoul, South Korea. ACM, New York, NY, USA

Analyzing Projects on GitLab

Process Metrics and Evolution

- See paper at *2021 IEEE Aerospace Conference, March 2021(to appear)*



Analyzing Software Engineering Processes with Provenance-based Knowledge Graphs

Andreas Schreiber
Institute for Software Technology
German Aerospace Center (DLR)
51147 Köln, Germany
andreas.schreiber@dlr.de

Lynn von Kurnatowski
Institute for Software Technology
German Aerospace Center (DLR)
82234 Weßling, Germany
lynn.kurnatowski@dlr.de

Claas de Boer
Institute of Software and Multimedia Technology
Dresden University of Technology
01062 Dresden, Germany
claas.deboer@dlr.de

Abstract—**Insights and assessments about the quality, reliability, or trustworthiness of software systems is important for many software applications. Especially for large or mission-critical software systems, reliable measures and assertions are crucial. Since software repositories contain information about source code, software development processes, and team interactions, we extract the provenance of software artifacts from those repositories and store the provenance according to a provenance model defined using W3C PROV data model. We use the recorded provenance to discover insights about the software and its development process, which we apply and evaluate for a large aerospace software system.**

TABLE OF CONTENTS

1. INTRODUCTION

Software is an important innovation factor and an essential part of modern research and development [1]. However, software development is a complex process, therefore software tools and technologies have been developed to help the software development process. All these support tools produce several types of data, which are generated before, during, and after the development of a software. These large amount of data can be explored by analyzing their *provenance*.

Today, many research fields use *provenance* [2] to verify data products and to analyze 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.

978-1-7281-7436-5/21/$31.00 ©2021 IEEE

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 provenance data model for software development processes. 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* and *retrospective provenance* and apply the methodology to the development processes for large aerospace software systems as follows:

- *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 provenance model for software development (Section 3).
- *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`-based development platforms, such as GITHUB or GITLAB (Section 4).
- We use *graph analytics* and *summarizing visualizations* on the retrospective provenance graphs. Especially, we present results for the space debris database system BACARDI (Section 5), which is hosted on the internal GITLAB of DLR.

2. PROVENANCE

Provenance can be expressed in many formats. We use the standardization recommendation World Wide Web Consortium (W3C) PROV [3], which defines the provenance data model PROV-DM [4] 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* (Figure 1) [5]. Each of the class elements and most relations can have additional *attributes*, which can further distinguish and characterize class elements and relations.

Use Case: Coronavirus "Contact Tracing Apps"

German "Corona Warn App" (CWA)

- App for *Exposure Notification*
- Based on APIs by Apple and Google
- Developed as **Open-Source Software** by SAP and Telekom
- External contributors (via pull requests)
- https://github.com/corona-warn-app
  - 13 repositories

Our Mission

- To analyze **the quality of CWA** and its **Open-Source development process**
- Generate advice for other government apps

- Sonnekalb, Tim und Heinze, Thomas S. und Kurnatowski, Lynn und Schreiber, Andreas und Gonzalez-Barahona, Jesus M. und Packer, Heather (2020) **Towards Automated, Provenance-Driven Security Audit for git-Based Repositories**. In SEAD 2020, 10.1145/3416507.3423190
- Schreiber, Andreas (2020) **Visualization of contributions to open-source projects**. In: 3th International Symposium on Visual Information Communication and Interaction. 10.1145/3430036.3430057

Tank you! Questions?

carina.haupt@dlr.de
@caha42