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 take 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
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

Recording of Data Processing Information

Application

Data (Results)

Provenance Store
Provenance Architecture at DLR

- Workflows
- Data & Metadata
- Software Development
- Provenance Store
- neo4j

Algorithms/Scripts
Machine Learning
Data Management
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

```
{
  "prefix": {
  }
}
```
Gitlab2PROV
https://github.com/DLR-SC/gitlab2prov

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

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")
Provenance Graph – Example Visualization with Graphviz/dot

Analyzing Software Engineering Processes with Provenance-based Knowledge Graphs

Andreas Schröder
Institute for Software Technology
German Aerospace Center (DLR)
51477 Köln, Germany
andreas.schroeder@dlr.de

Lynn von Kurnatowski
Institute for Software Technology
German Aerospace Center (DLR)
52324 Wessling, Germany
lynn.vonkurnatowski@dlr.de

Abstract—Insights and monstum-s about the quality, reliability, or transferability of software systems are important for many software applications. Especially for large or mission-critical software systems, reliable measures and assertions are crucial. Therefore, software development processes, and their interactions, are essential for the provenance of software artifacts from the development repositories and over the provenance according to a provenance model data is a prerequisite for the model’s PARV data model. We use the recorded provenance to observe insights about the software and its development process, which we apply and evaluate for a large aerospace software system.

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 with the development process. This process has several types of data, which are generated, during, and after the development of a software. These large amounts of data can be exploited by analyzing their provenance.

Today, many research fields use provenance [2] to verify data products and to analyze processes that led to the provenance. Provenance can be used to form assessments about quality, reliability, or transferability of a piece of data. The knowledge of provenance includes aspects such as sources and processing steps as well as dependencies and contextual information.

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-OM [4] to support the interoperable interchange of provenance in heterogeneous environments such as the web.

The core structure of PROV-OM relies on the definition of the model class elements relations, attributes, and links that are involved in producing a piece of data or artifact and on definitions of relations to relate these class elements, such as wasGeneratedBy, wasSourceWeb, and redefinitionOf, and used (Figure 1-3). Each of the class elements and most relations 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](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

Tank you! Questions?
carina.haupt@dlr.de
@caha42

#WIRROCKENSOFTWARE