# Data Provenance @ 2nd DAMe Workshop

André Giesler | JSC@Forschungszentrum Jülich GmbH

# Content

- Part 1 - Overview ( ~ 60 minutes)
  - Introduction
  - Provenance Capturing
  - Modeling and PROV standard
  - Provenance Storage and Querying
  - Some useful provenance tools
- Part 2 - Hands-on-Session ( ~ 90 minutes)
  - Exemplary work with provenance tools in 2 exercises

# Introduction to Provenance

# Provenance – Etymology

- from latin: "provenire"
- the place of **origin** or earliest **history** of something
- the **beginning** of something's existence
- in art history a record of ownership of a work of art
- used as a guide to authenticity or quality
- related terms: lineage, genealogy, traceability, …

1434: painting dated by van Eyck;
before 1516:in possession of Don Diego de Guevara, a Spanish career courtier of the Habsburgs;
1516: portrait given to Margaret of Austria, Habsburg Regent of the Netherlands;
……
1816: in London, probably plundered by a certain Colonel James Hay after the Battle of Vitoria (1813), from a coach loaded with easily portable artworks by King Joseph Bonaparte;
1841: the painting was included in a public exhibition;
1842: bought by the National Gallery, London for £600, where it remains.

# Areas of Application for Provenance

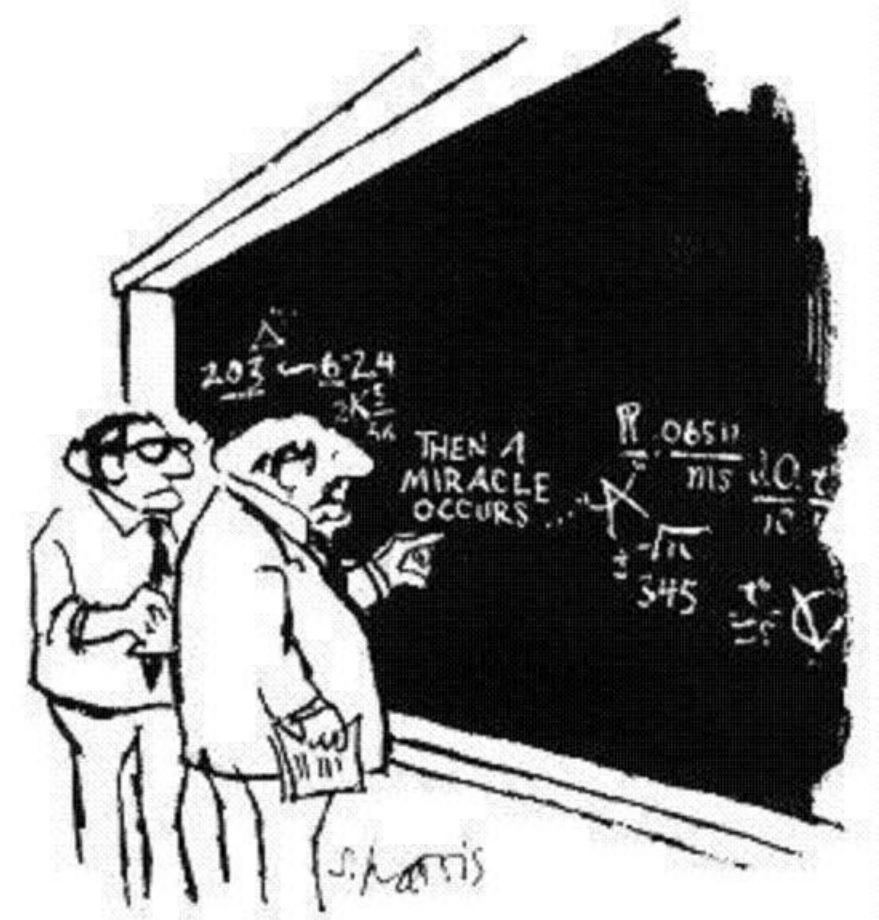
- News
  - origins and references of blogs, photos, news items
- Copyright Laws
  - licensing and attribution of documents and data
- Food industry
  - trust in supply chains
- Open Information System
  - origin of the data inputs
- **Computational Provenance**
  - **how the results were obtained**

# Computational Provenance - Definitions

- Simple: “Origin and processing history of an artifact”
- Formal: “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.”
- Another: “Data provenance/lineage includes  data’s origins and what happens to it over time”

# Reproducibility/Replicability

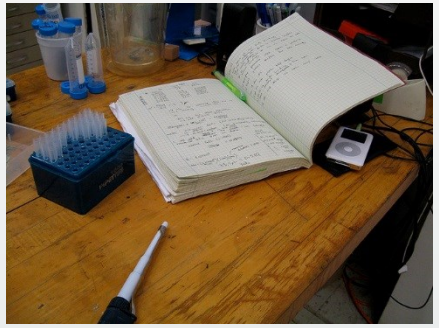
- Terms (Rougier et al. 2017)
  - Reproducibility: Running same software on same input and obtaining same results
  - Replicability: Writing and running new software based on description of a computational model …. and obtaining results that are similar enough
- Reproducibility in data science is based on
  - Open Source Software
  - Code Repositories
  - Publications with code
  - Virtual machines / containers
  - (Electronic) laboratory notebooks
  - **Metadata and Provenance**




"I think you should be more explicit here in step two."

# Reproducibility Crisis in Science

- results of many scientific studies are difficult/impossible to reproduce https://doi.org/10.1038%2F515009a
- Reasons
  - correctness of implementation
  - potential for error introduced by new system soft- and hardware
  - how precise one had performed an experiment
  - time aspect – difficult to construct after years – or even weeks



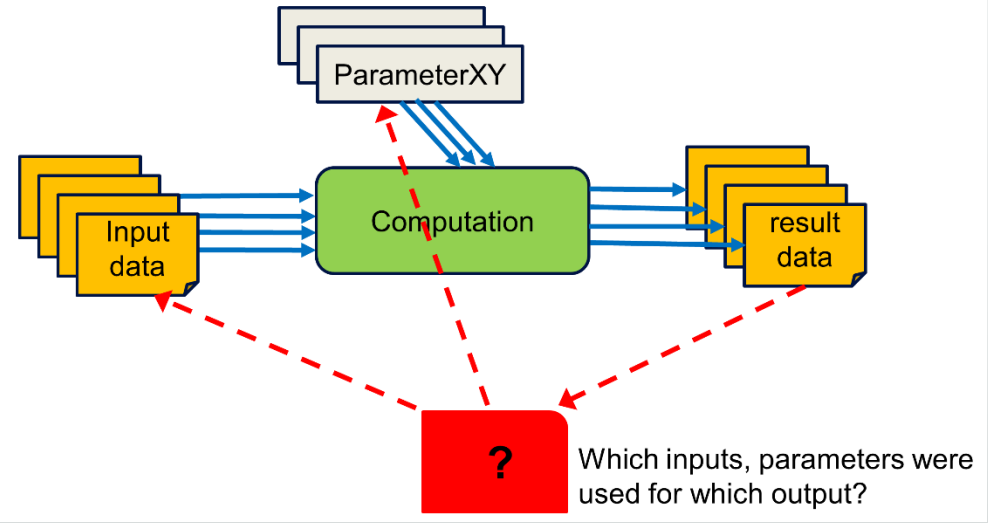
- reproducibility of experiments is an essential part of the scientific method
  - significant theories are grounded on unreproducible experimental work
- Scientific research is held to be of good provenance
  - if documented in detail
  - sufficient to allow reliable replication of scientific results

# Use Cases for Computational Provenance

- Audit trail
  - trace data generation and detect possible errors
- Attribution
  - determine ownership and responsibility for data and scientific results
- Data quality
  - from quality of input data, computations
- Discovery
  - enable searching of data, methodologies and experiments
- Replication:
  - facilitate repeatable derivation of data
  - allowing reproducibility in science

# In Practice…

- Do any of these sound familiar?
  - I thought I used the same parameters but I'm getting different results
  - I can't remember which version of the code I used to generate figure 6
  - The new student wants to reuse that model I published three years ago but he can't reproduce the figures
  - It worked yesterday
  - Why did I do that?



# Provenance Data/Information

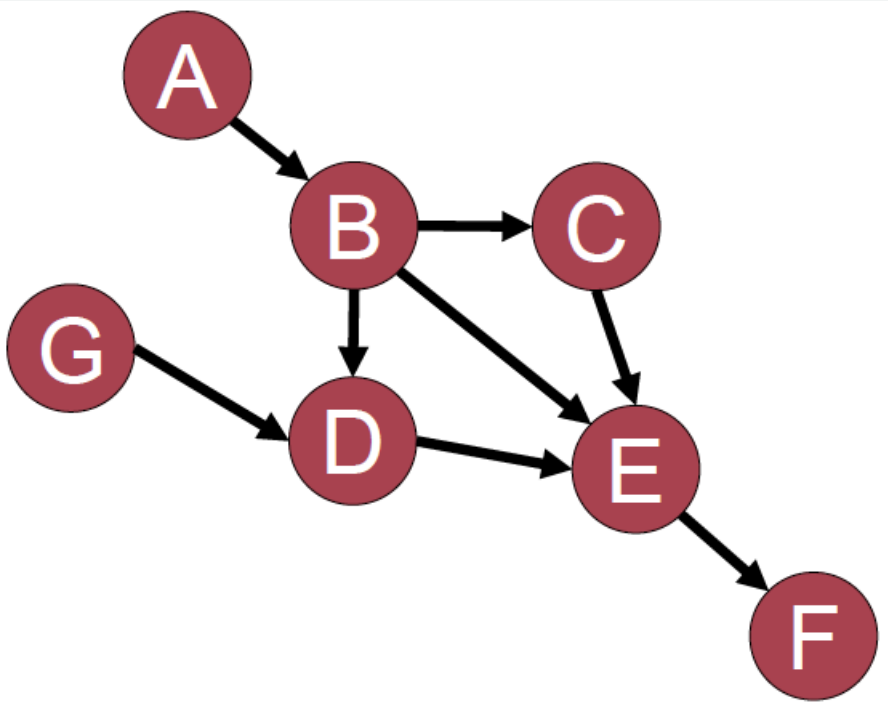
- What is Provenance Data?
  - The source of all input data (references)
  - The set of the algorithms and scripts used to transform the data
  - A record of single compute job on an HPC machine with inputs and outputs
  - A complete workflow description consisting of many compute jobs
  - The complete description of the processing environment

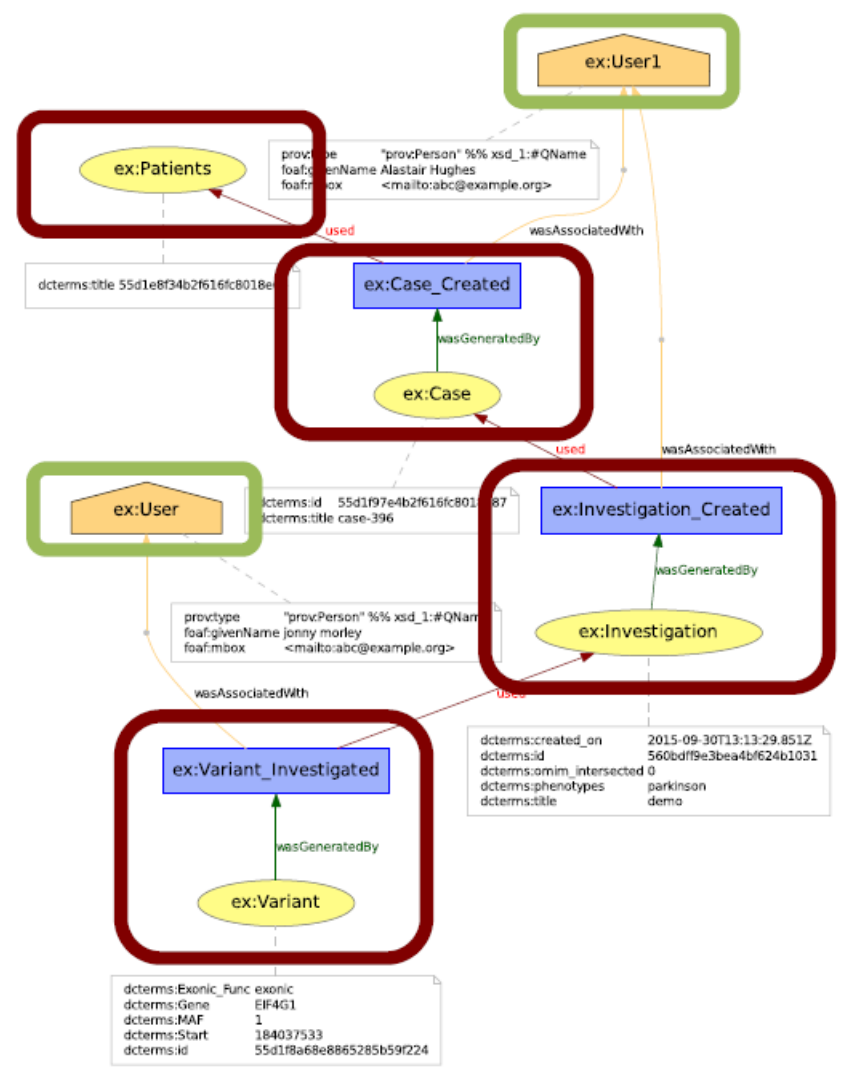
# Provenance (data) and Metadata

- **Data**: Actual used, measured, simulated, or generated data
- **Metadata**: Information about the resources itself (descriptive terms)
- **Provenance**: Metadata with a focus of Who? (e.g. creator), When? (e.g. modified when), and How? (e.g. isDerivationFrom)
- Provenance is Metadata, but not all Metadata is important for Provenance
  - the title or format of a book is not part of its provenance
  - however, who created the book (author and publisher), or when was it created
  - Provenance creates a semantic graph of the data history
- In practice Metadata and Provenance is mixed up
  - when you track the provenance, you would also track the metadata
  - keep it in the same storage

# Provenance is a Graph

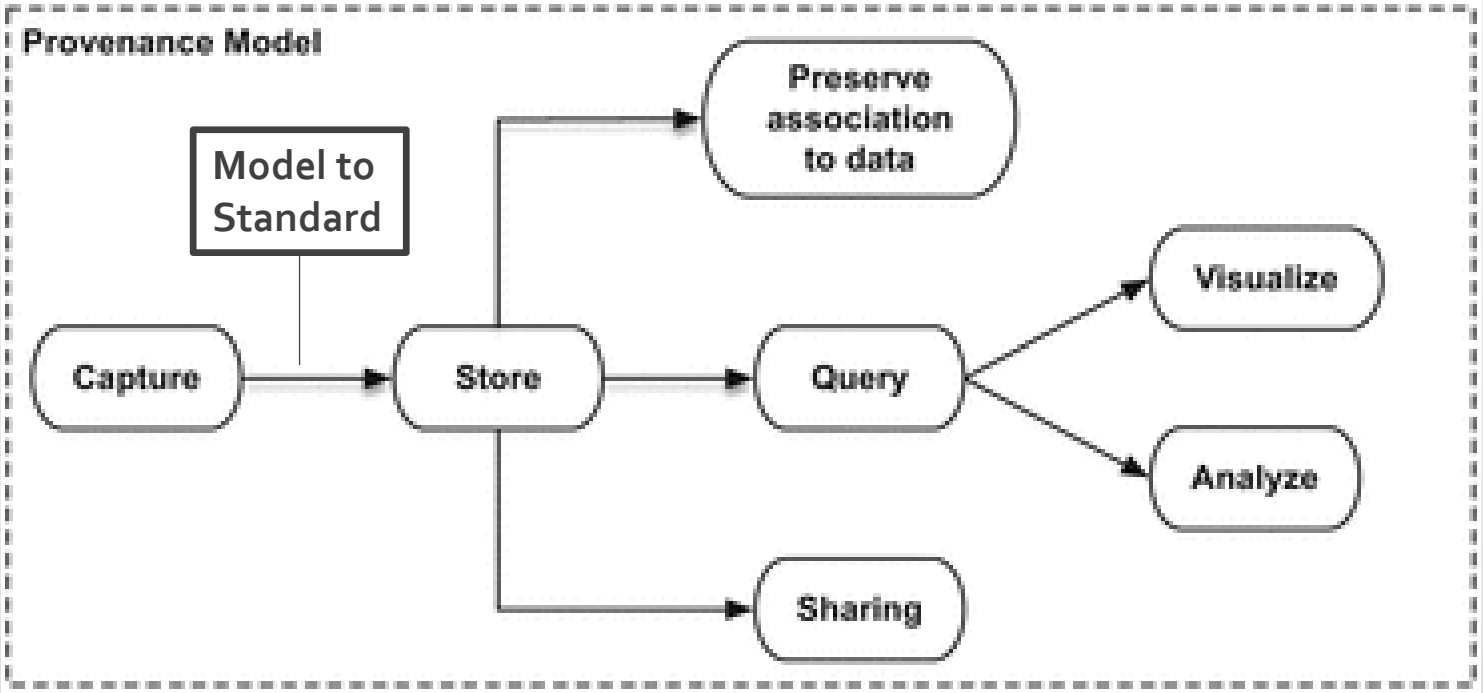
- Directed Acyclic Graph (DAG)



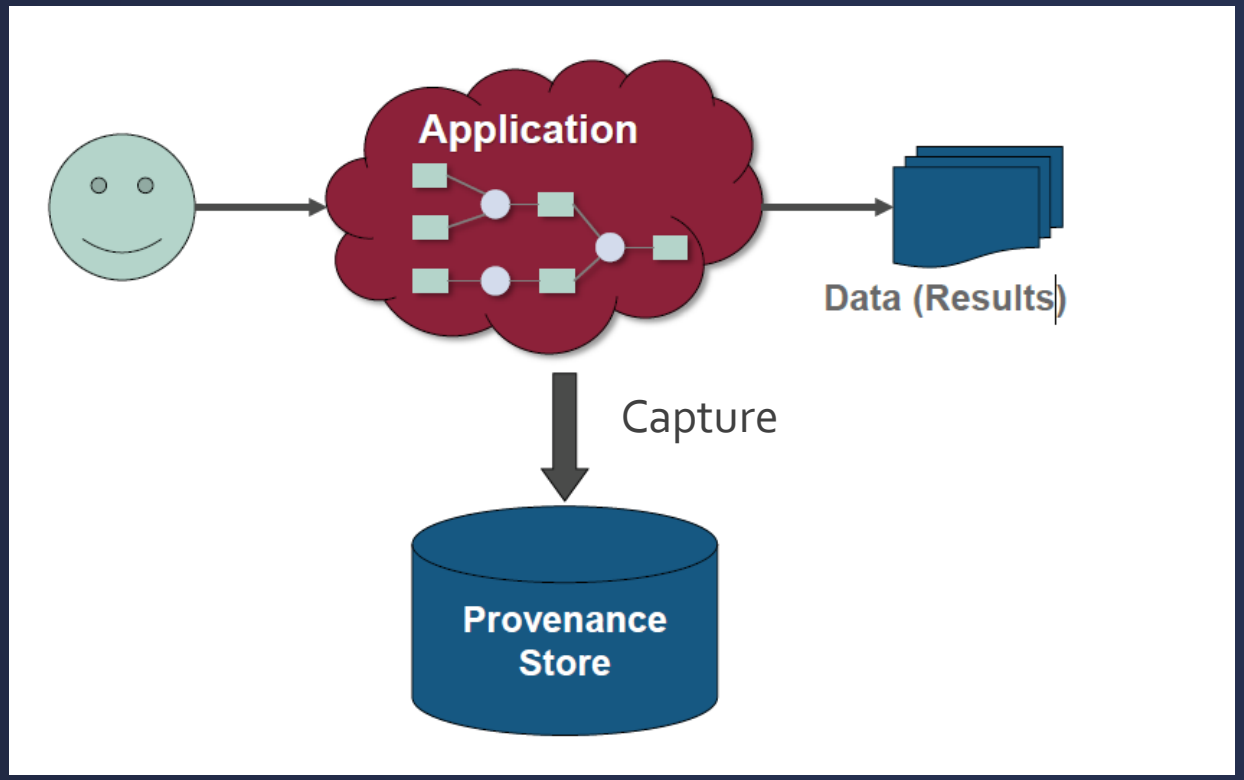



# Provenance Lifecycle

- Schema of Provenance management



# Capturing Provenance



# Capturing Provenance - Challenges

- Provenance is result of observing transformation processes in environments
  - Observation depends entirely on the available infrastructure
- retrofit applications and systems with provenance recording capabilities
- Different researchers have very different ways of working and use different tools, workflows, storages etc.
  - Environments: command line, GUI, batch-jobs, workflow engines or scripted workflows
  - Languages for data analytics: Python, R, …
- Level of abstraction of provenance information
  - fine-grained – e.g. each I/O operation in an observed system
  - coarse-grained – only capturing inputs and outputs of a workflow block
- Distributed systems – captured data must be composed from multiple fragments
- Overhead - should have minor impact on system/applications performance
- Scalability – Big data is growing -> provenance is growing -> scale up provenance collection

# Provenance Capturing Systems – Application Level

- Enable applications to be able for provenance capturing
- Provide often rich semantic knowledge of provenance data
  - Scientist and developer is often identical
- Workflow engines are suitable for provenance capturing
  - bring along already logging mechanisms
  - tracking of inputs and outputs of each workflow step
  - dependencies (sequential, parallel) between workflow elements
- Make often use of domain user annotation
- disadvantages
  - users are restricted of using provenance-aware application
  - responsibility of tracking provenance is dispersed throughout many different tools

# Provenance Capturing Systems – System Level

- System-level provenance capturing aka Automated provenance observing at OS system
- requires minimal or no modification to given applications
- reduce the need of re-engineering software
- capture complete provenance as instrumentation can be done
  - broadly – across every application
  - deeply – within specified part of file system
- disadvantages
  - can capture only provenance to which the system is exposed (e.g. observed by OS: FUSE, Linux Audit, etc…)
  - produces a lot provenance with less semantic meaning
- In practice, capturing on system level still needs more research

# Provenance Capturing Systems – VCS Level

- Version Control Systems (VCS) facilitate collaboration primarily for both code and data
- Tracking provenance through the use of VCS such as Git
- Exposing provenance information stored in VCS
- Drawbacks
  - VCS suppose to have many small files with localized changes
  - data science applications typically feature a range of large datasets -> suboptimal behaviour
  - VCS systems have limited support for provenance queries
- Implementations: Sumatra (hands-on-session)

# Gather or Generate Provenance

- Depends on your application
- Gather at runtime (e.g. tracked by logging of workflow engine)
  - Tailoring the application
  - Cumbersome from software engineering perspective
  - Combined with logging, reconstruction from files and metadata
  - Also known as **Retrospective Provenance**
- Generate at compile time (e.g. scripted workflows without observing engine)
  - Based on static code analysis
  - Adding special statements to existing scripts
  - Also known as **Prospective Provenance**

# Other Techniques for Provenance Recording

- Provenance APIs
  - PyPROV Python API (Hands-on session)
  - Sumatra API
- Wrapping applications for Provenance
  - NoWorkflow (performs code analysis and extracts provenance from scripted workflows)

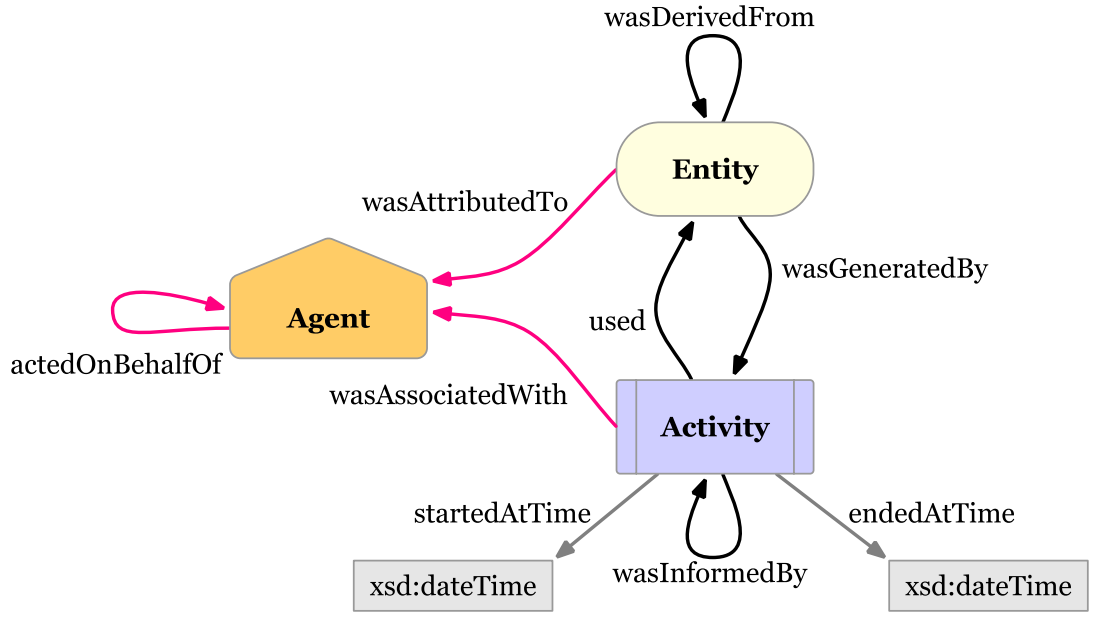
# Modeling and PROV standard

# Provenance Standard Models

- Why taking a common provenance model?
  - Having a common provenance language
  - Inter-changing provenance data between software and humans (readability)
  - Providing standard interfaces in provenance tools
- Obstacles
  - Map domain data / captured data at first to standard model
  - Extend standard model where applicable
- Some efforts to introduce an accepted world-wide standard
  - OPM
  - and successor W3C-PROV (since 2013)

# W3C PROV

- Published as a 'PROV' family of recommendations in 2013

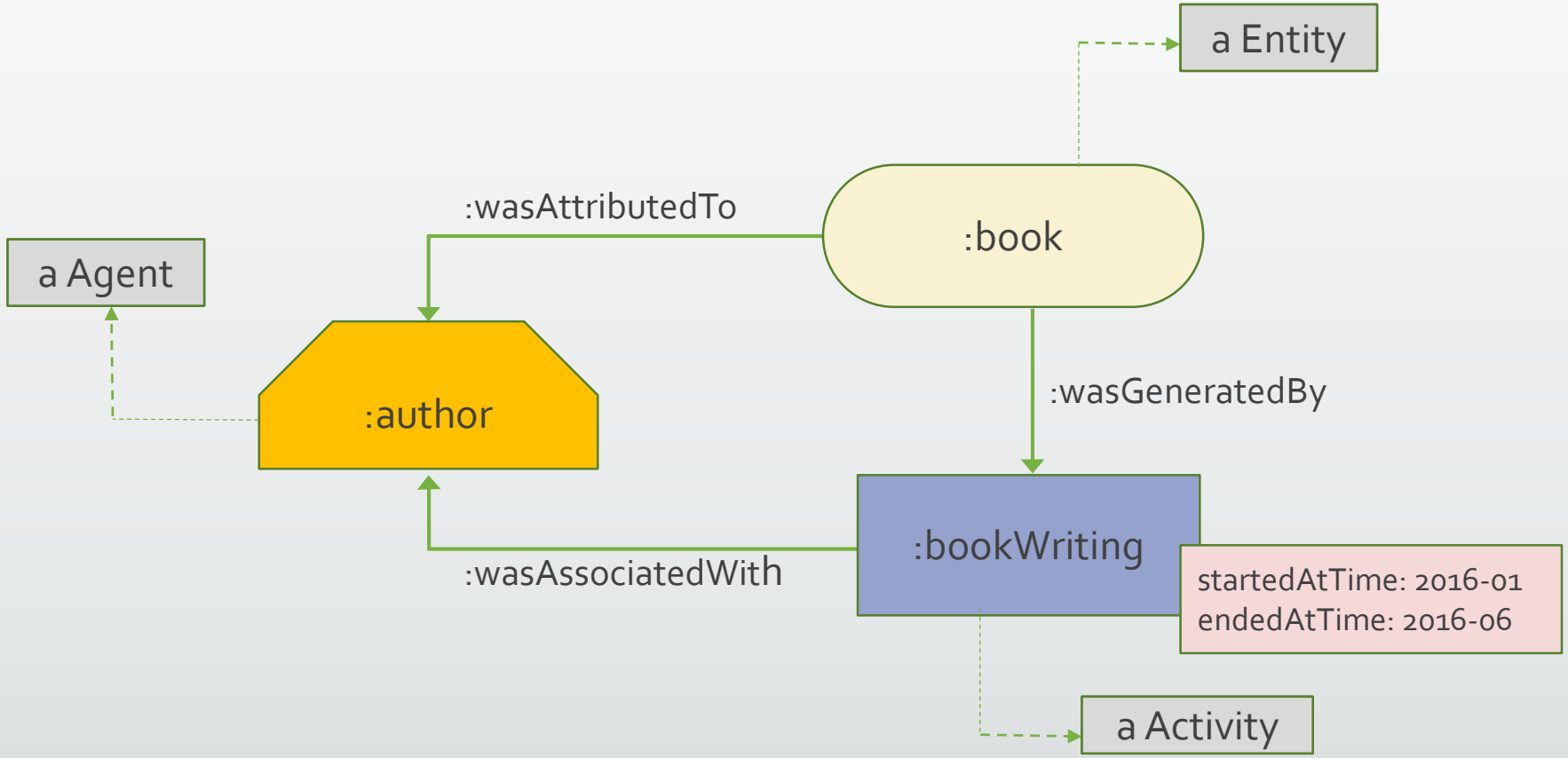



Provenance is a record that describes the people, institutions, entities, and activities, involved in producing, influencing, or delivering a piece of data or a thing in the world.

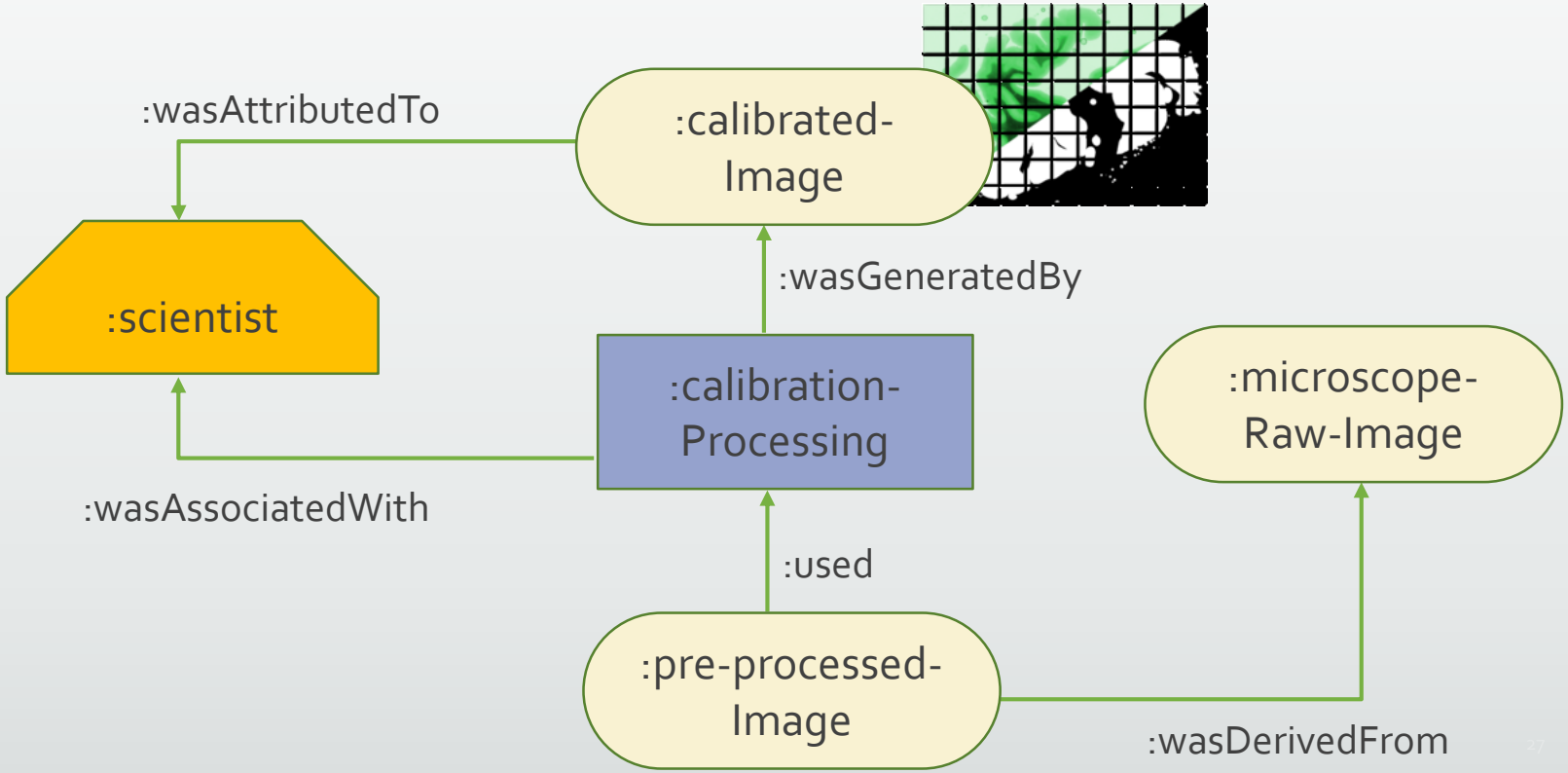
# PROV: Fundamental Notions

- *Entity*
  - the "things" whose provenance we want to describe (physical, digital, or other)
  - e.g. document, image, data file, program code…
- *Activity*
  - generate new or use existing entities. The "dynamic" aspect of the world
  - e.g. program execution, image processing, …
- *Agent*
  - are responsible for the actions (persons, organizations, software agent,…)
- Relations as *usage*, *generation*, *derivation*, *attribution*
  - connections describing how entities, agents, and activities interact

# Simple PROV Example – Book Writing



# Scientific PROV Example – Image Processing



# How to work with the PROV model?

- Universal core data model (PROV-DM) in textual form
- Various Serializations and definitions available
  - PROV-O OWL2 ontology
  - PROV-XML defines XML schema for PROV-DM
  - PROV-N human-readable special provenance notation
  - PROV-JSON(-LD) lightweight extensible representation
- Can be specialized/extended to model provenance information for
  - different domains and vocabularies (biology, climate research, etc.)
  - applications (e.g. workflow management systems)

# Ontologies and Provenance Models

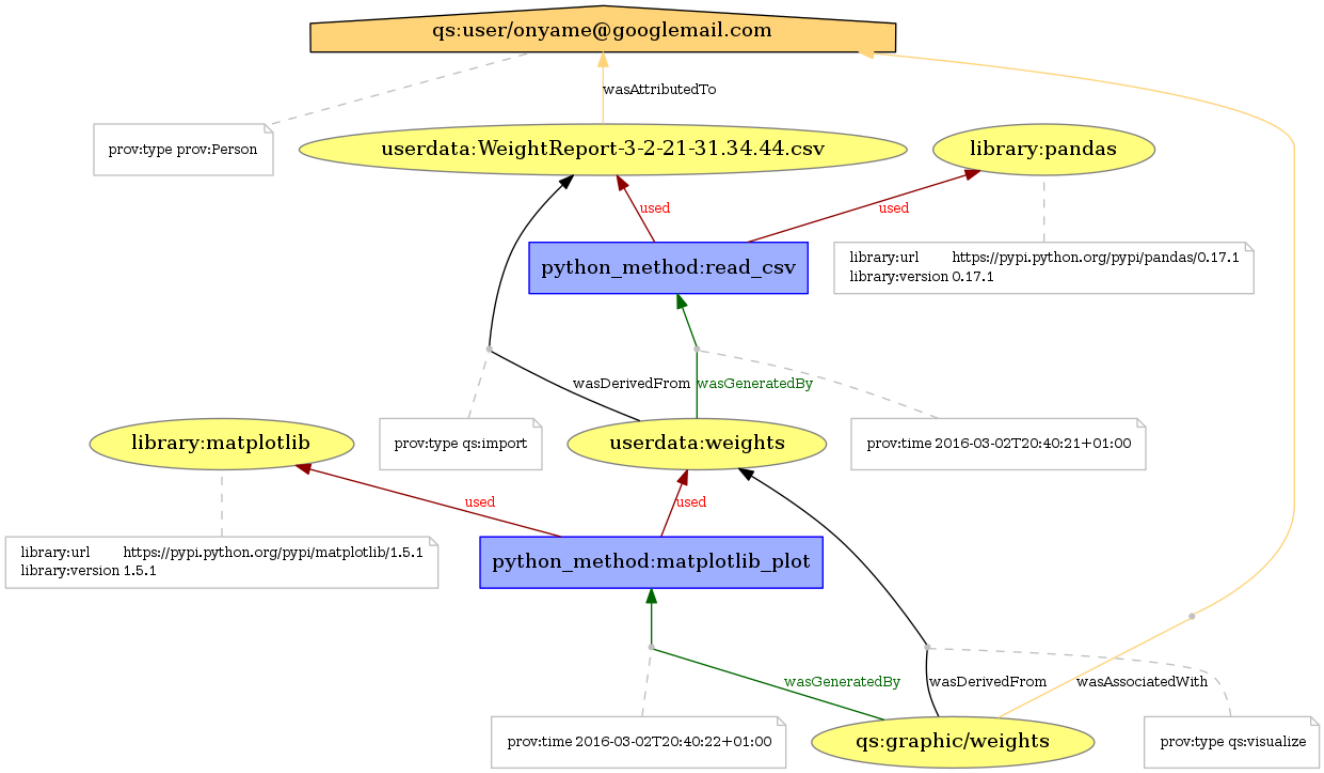
- Close connection
- Ontologies can express Provenance models perfectly
  - PROV-O is ontology serialization of W3C PROV data model
  - Defining domain terms in ontologies by extending PROV-O
  - Noted in OWL and RDF like PROV-O
- Integration in applications by generating complement classes/ code
  - e.g. in Java by using Apache Jena RDF and ontology framework
  - e.g. in Python by Owlready2 or ontology-python-sdk
- Applications can create syntactically PROV (-extended) conform provenance data

# PROV Notations and Representations

- Textual Representation (e.g. PROV-N, RDF-Turtle, XML, JSON)

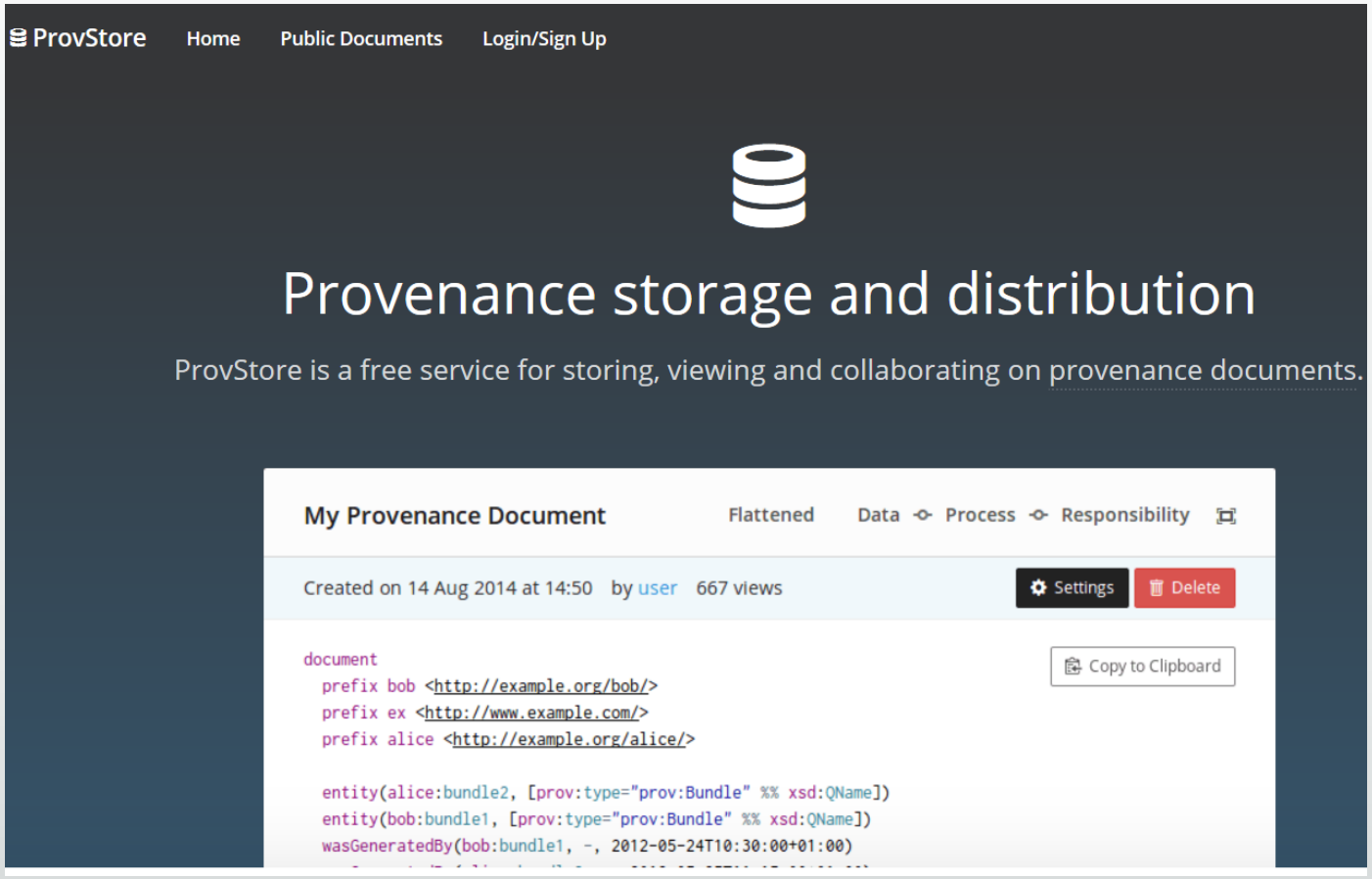
```
document
prefix userdata http://software.dlr.de/qs/userdata/
. . .
wasDerivedFrom(userdata:weights, userdata:WeightReport.csv,
wasDerivedFrom(qs:graphic/weights, userdata:weights,
wasAssociatedWith(qs:graphic/weights, qs:user/onyame@gmail.com, -)
used(python_method:read_csv, library:pandas, -)
used(python_method:matplotlib_plot, 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
```

- Visualization



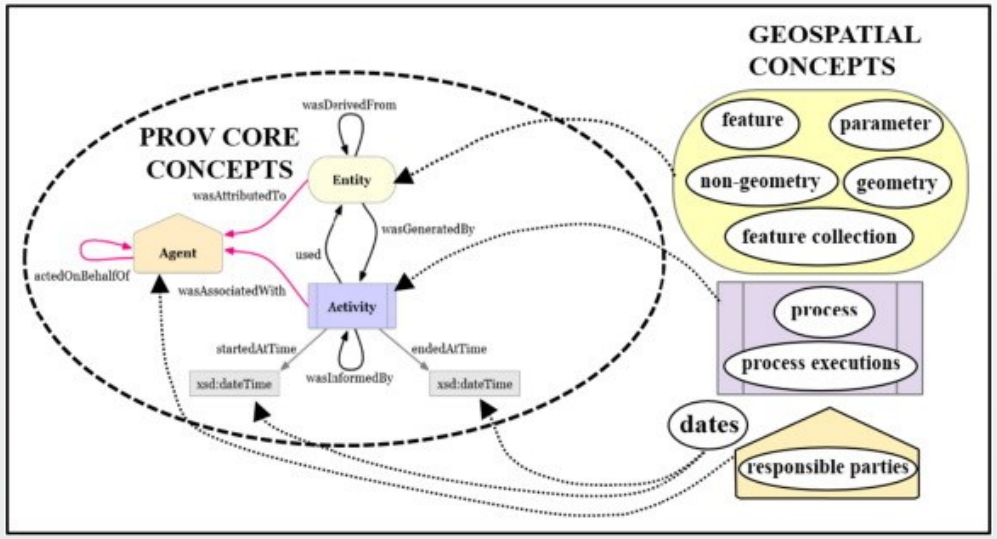
# Cloud Services related to PROV

- PROVStore of University of Southampton (openprovenance.org)
  - Store PROV documents
  - Share
  - Validate
  - Translate / Convert
  - Visualize
  - APIs (Python, jQuery)



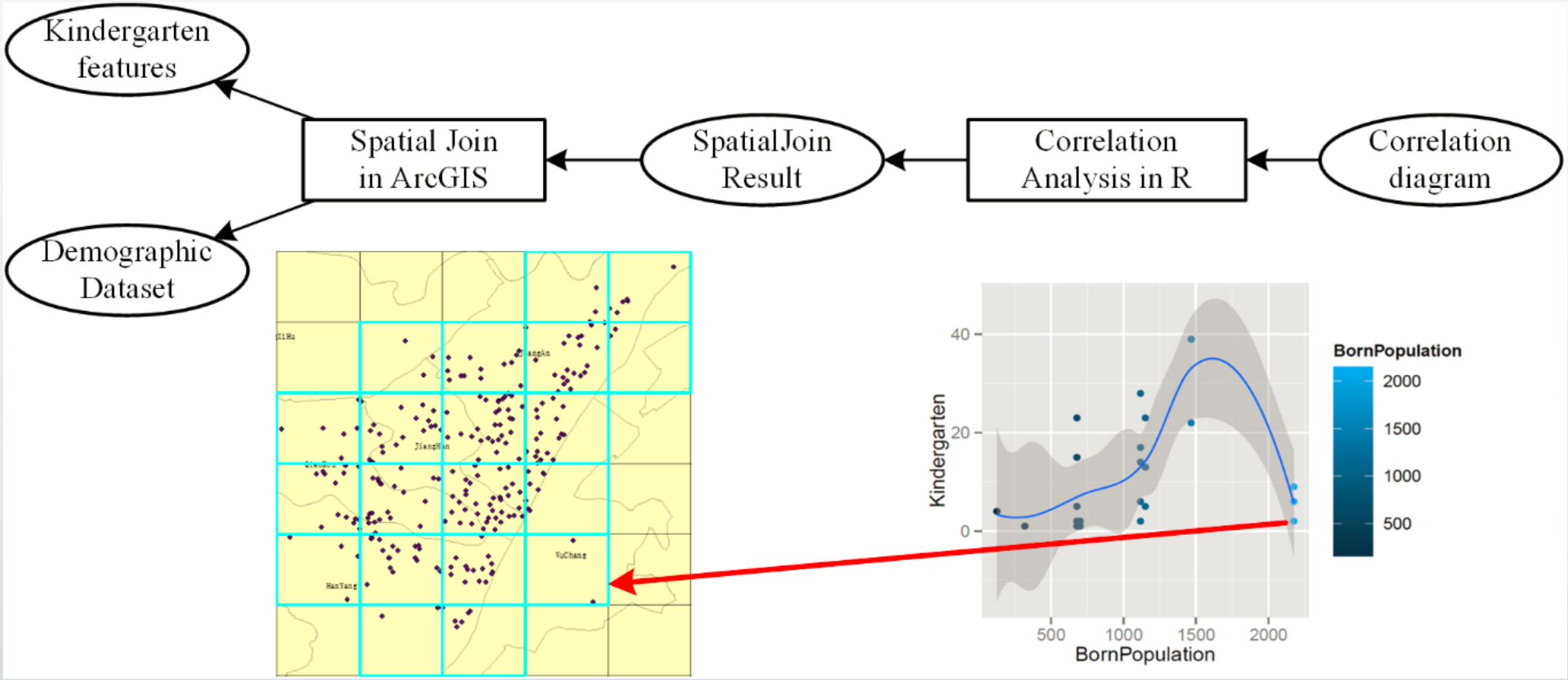
# PROV Implementations: Geospatial Data I

- PROV application in geospatial environments requires matching process with ISO19115 data
- GeoProv: Mapping ISO19115 lineage terms to W3C PROV [Closa et. al. 2017]
- Tracing back origins of geospatial data




```
...
<rdf:Description rdf:about="http://geos.whu.edu.cn/ont/iso19115/lineage.owl#LI_ProcessStep">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/ns/prov#Activity"/>
</rdf:Description>
...
```

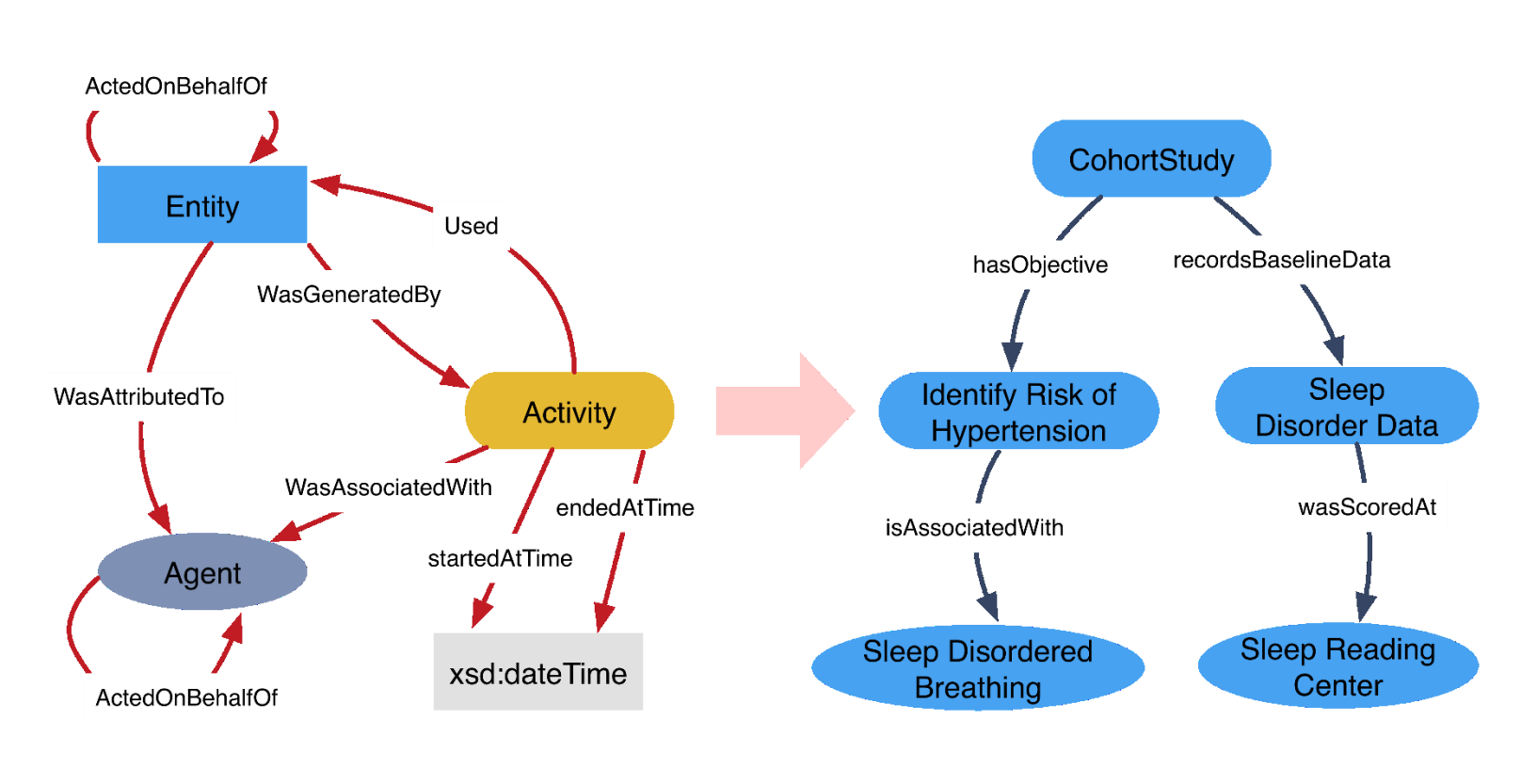
# PROV Implementations: Geospatial Data II




[https://github.com/liangcun/GeoProv]

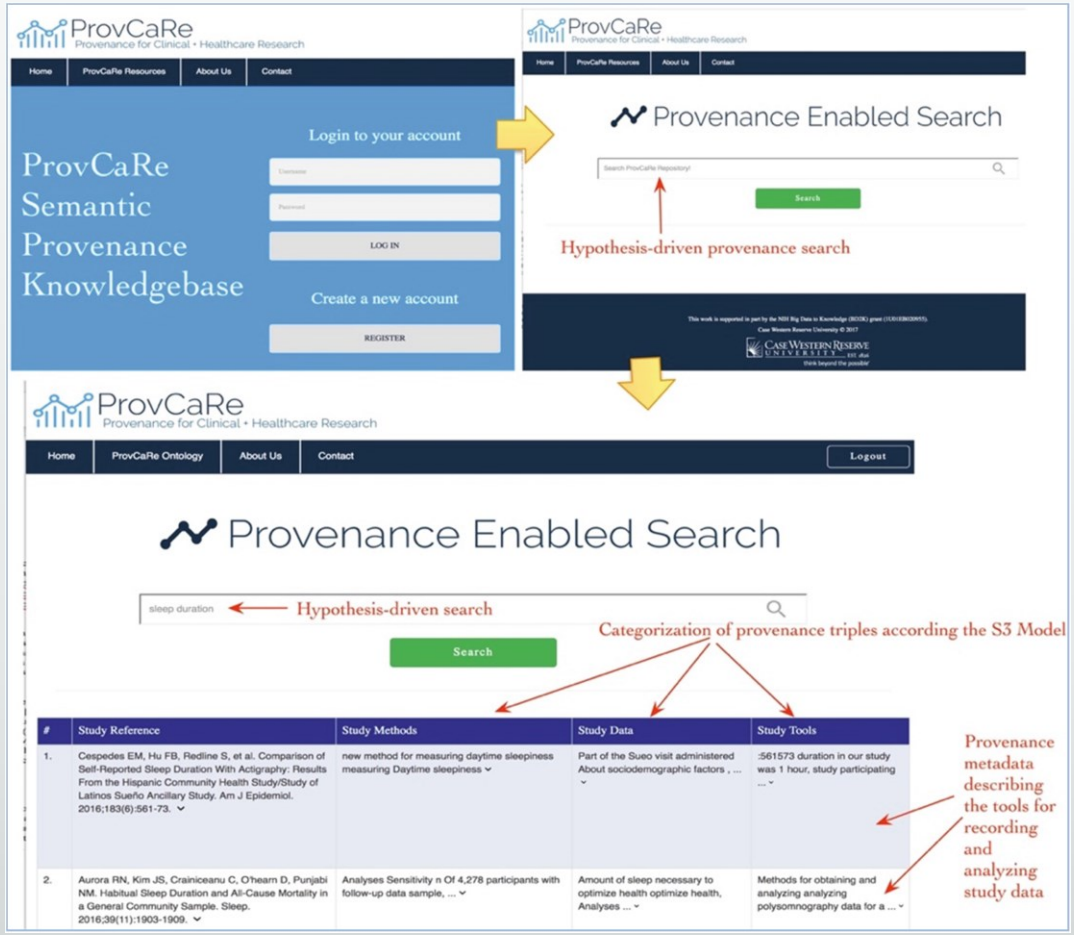
# PROV Extensions: Biomedical Data I

- Provenance for Clinical and Healthcare Research (ProvCaRe) supports modeling, querying and analysis of semantic provenance for reproducing biomedical research studies
- ProvCaRe extends PROV-O to text-process biomedica studies and extrac provenance graph from published articles



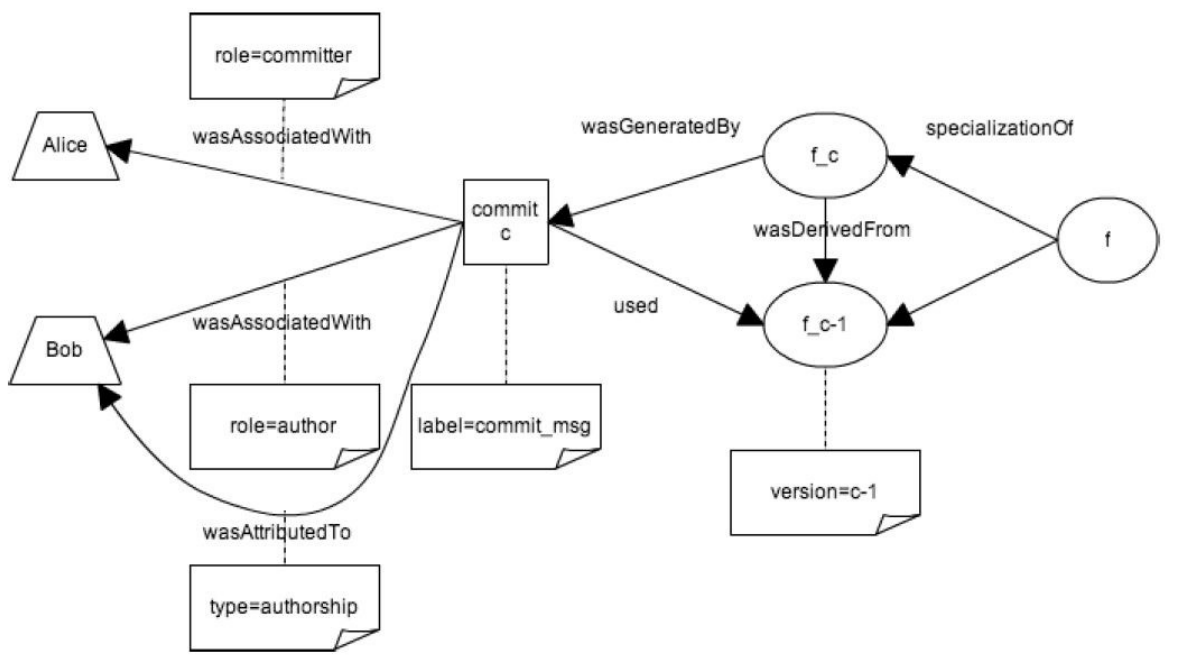
# PROV Extensions : Biomedical Data II

- Provenance Knowledge Repository: consisting of provenance information extracted from published research studies
- can be queried via browser interface
- https://provcare.case.edu/



# PROV Extensions : Version Control Systems

- VCSs are one mechanism to track provenance (part of it)
- Mapping Git operations to PROV
- Interchanging provenance information of GIT
- Goal – publication of VCS provenance and subsequent integration with other PROV enabled systems
- git2prov.org



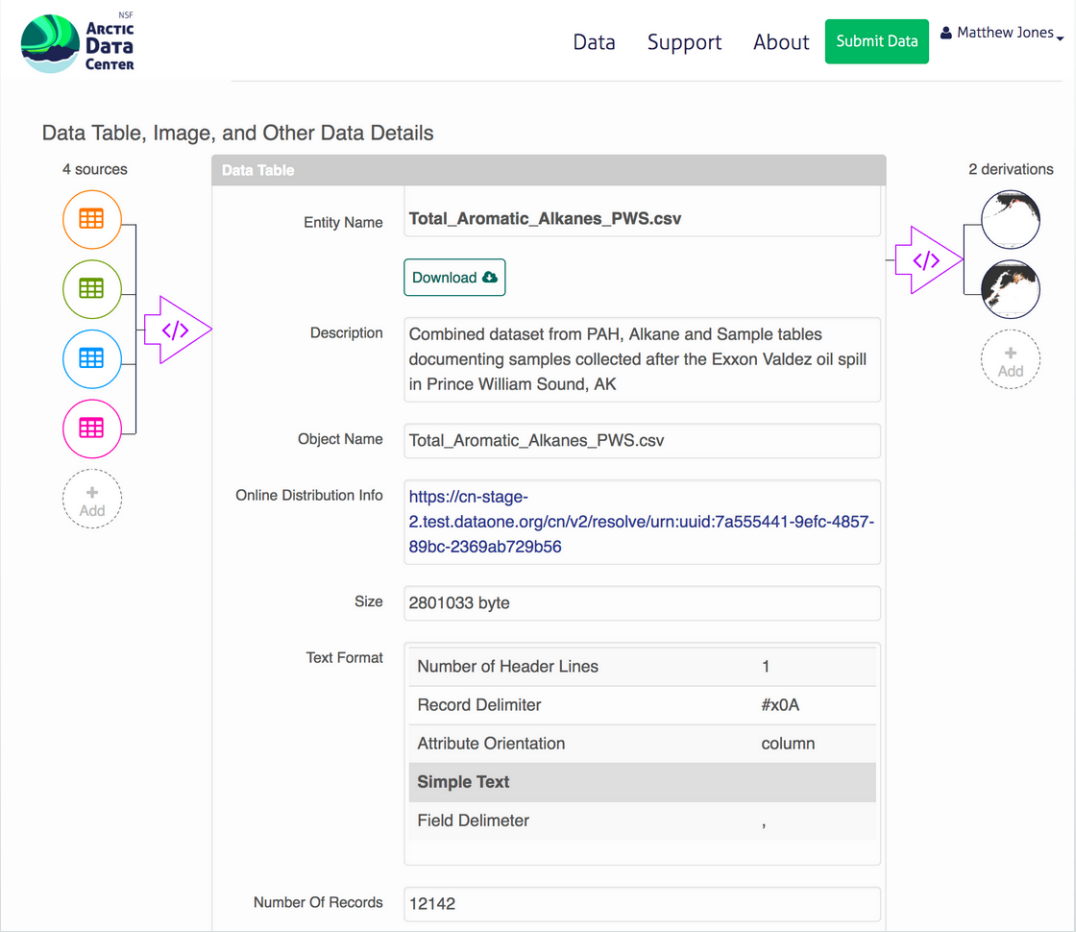
# PROV Extensions : Dublin Core Mapping

- DC provides core metadata vocabulary for generic resource descriptions
- PROV is focused on expressing actions and resource states in a provenance chain
- Mapping between DC and PROV to:
  - bridge the gap between DC and PROV
  - help to derive PROV data from large amount of existing DC data
- https://www.w3.org/TR/2013/NOTE-prov-dc-20130430/

| DC Term | Relation | PROV Term | Rationale |
|---|---|---|---|
| dct:Agent | owl:equivalentClass | prov:Agent | Both `dct:Agent` and `prov:Agent` refer to the same concept: a resource that has the power to act (which then has responsibility for an activity, entity or other agent). |
| dct:BibliographicResource | rdfs:subClassOf | prov:Entity | A bibliographic resource refers to books, articles, etc., which are concrete PROV entities. |
| dct:LicenseDocument | rdfs:subClassOf | prov:Entity | Document granting permission to do something regarding a resource. Thus, it is mapped as a type of `prov:Entity`. |

# PROV Extensions : ProvONE for Workflows

- Developed in the context of the DataONE project
- Workflow Management Systems (WfMS) generate special provenance of
  - the model specifying the workflow themselves (prospective provenance)
  - the traces from the workflow executions (retrospective)
  - the dataflow of inputs, intermediate results, and resulting outputs
- PROV alone do not suffice for encoding scientific workflow provenance
- ProvONE extension is designed to support a large variety of WfMSs (e.g. UNICORE)



# Provenance Storage

# Provenance Storage

- Storing, indexing and querying provenance requires a data layer
- Bound to have an impact on data volume
  - depending on observed source data and granularity of provenance data
  - backup option for preserving provenance data
- Provenance data and documents are naturally expressed in form of a graph
  - suggests the usage of graph databases (GDBMS) e.g. Neo4J
  - nevertheless relational database solutions often used (RDBMS)
  - driven by standardization of provenance in RDF -> triple stores
  - alternative – storing provenance documents in file system (e.g. querying by grep)

# Provenance Storage – Relational DBMS

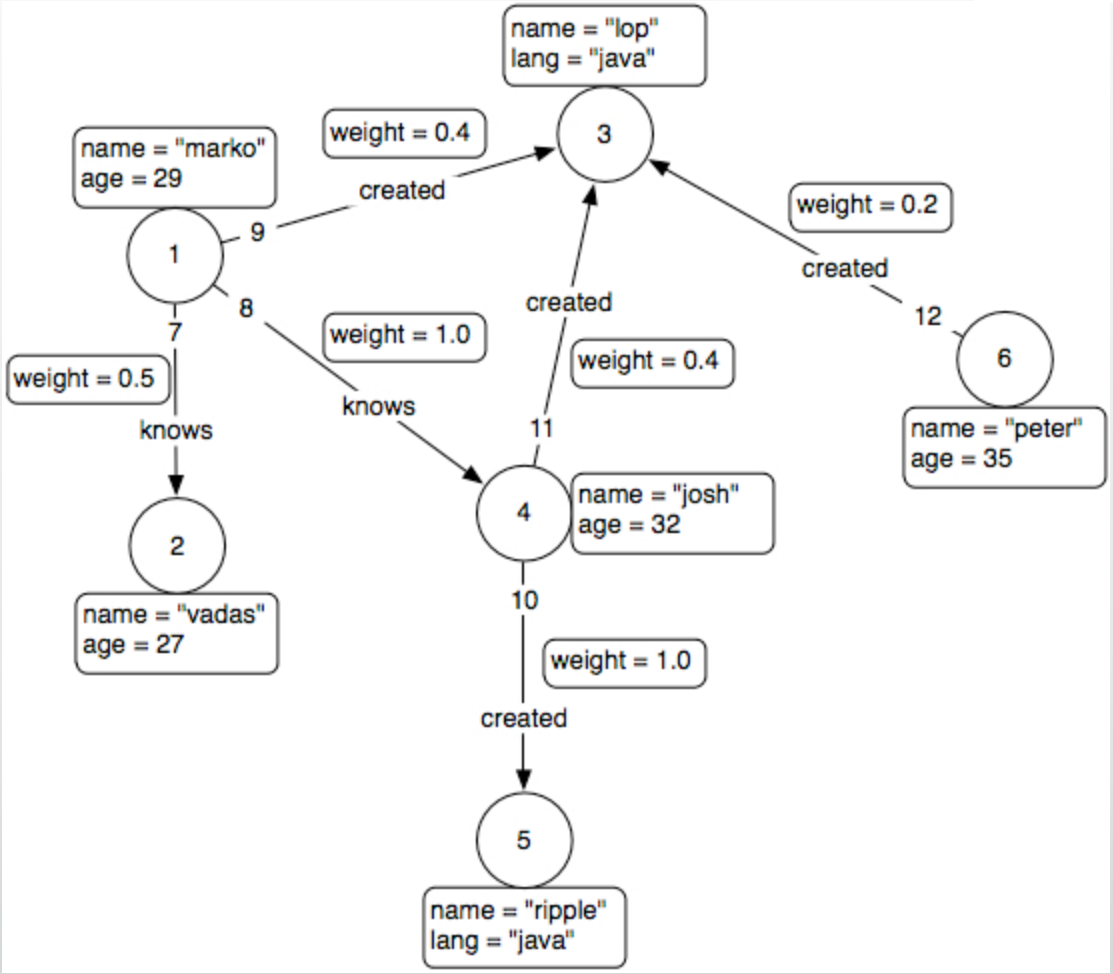
- Attractive option when
  - when provenance data has high level of abstraction (short paths)
  - if captured data is already organised as structured data (ERM model)
  - relational operators support query graphs
  - SQL infrastructures and experience is already existent
- Drawbacks
  - construct graph queries as SQL queries (repeated self-joins to traverse the path)
  - addition of new attributes results in changes to relational schema
  - these limitations contribute to performance degrading as provenance graphs grew

# Provenance Storage – Graph Databases

- Natural representation of provenance graphs
- Recent provenance projects make use of GDBMS (e.g. Neo4J)
- Simple graph traversing along the graph by using native query languages
  - e.g. Cypher in Neo4J
- Attractive option when
  - data is already organised as semi-structured data (e.g. XML)
  - complex traversing along the provenance graph is required
  - provenance data has been tracked from complex sources (graphs with many vertices and edges)
  - nature of provenance information is  frequently changing (no schema required in GDBMS)
- Drawbacks
  - less performance when have queries like "Give me highest value from this key"

# Provenance Storage – Neo4j GDBMS

- Open-Source
- Implemented in Java
- Stores property graphs (key-value-based, directed)
- Able to consume RDF graphs directly with Neo4j-extension
- http://neo4j.com



# Provenance Storage – Store and Retrieve

| Storage Concept | Query Mechanism |
|---|---|
| Relational databases | SQL |
| XML | Xpath |
| RDF / Triple-Stores | SPARQL |
| Graph databases | e.g. Cypher |

- Reasonable for users, non-experts?
- How to avoid that users have to learn query languages?
  - Web interfaces
  - Enable full-text search

- Cloud Services
  - ProvStore - https://openprovenance.org/store/
  - Suitable for testing, data interchange

# Provenance Storage – Further Aspects

- Preserving references of observed data in Provenance data
  - What happens if the referenced data is moved to another repository?
  - Sync references or make use of PIDs?
- Provenance Analytics
  - all forms of consumption and exploitation of provenance content
  - what can be learned from a large body of provenance metadata?
  - what techniques and algorithms can be successfully borrowed from data analytics
- Provenance vs. Privacy and Security
  - Unauthorized parties should not have access to provenance records or parts of it

# Provenance Tools and Projects

- Exemplary selection of tools
- Different technique
- Different approach

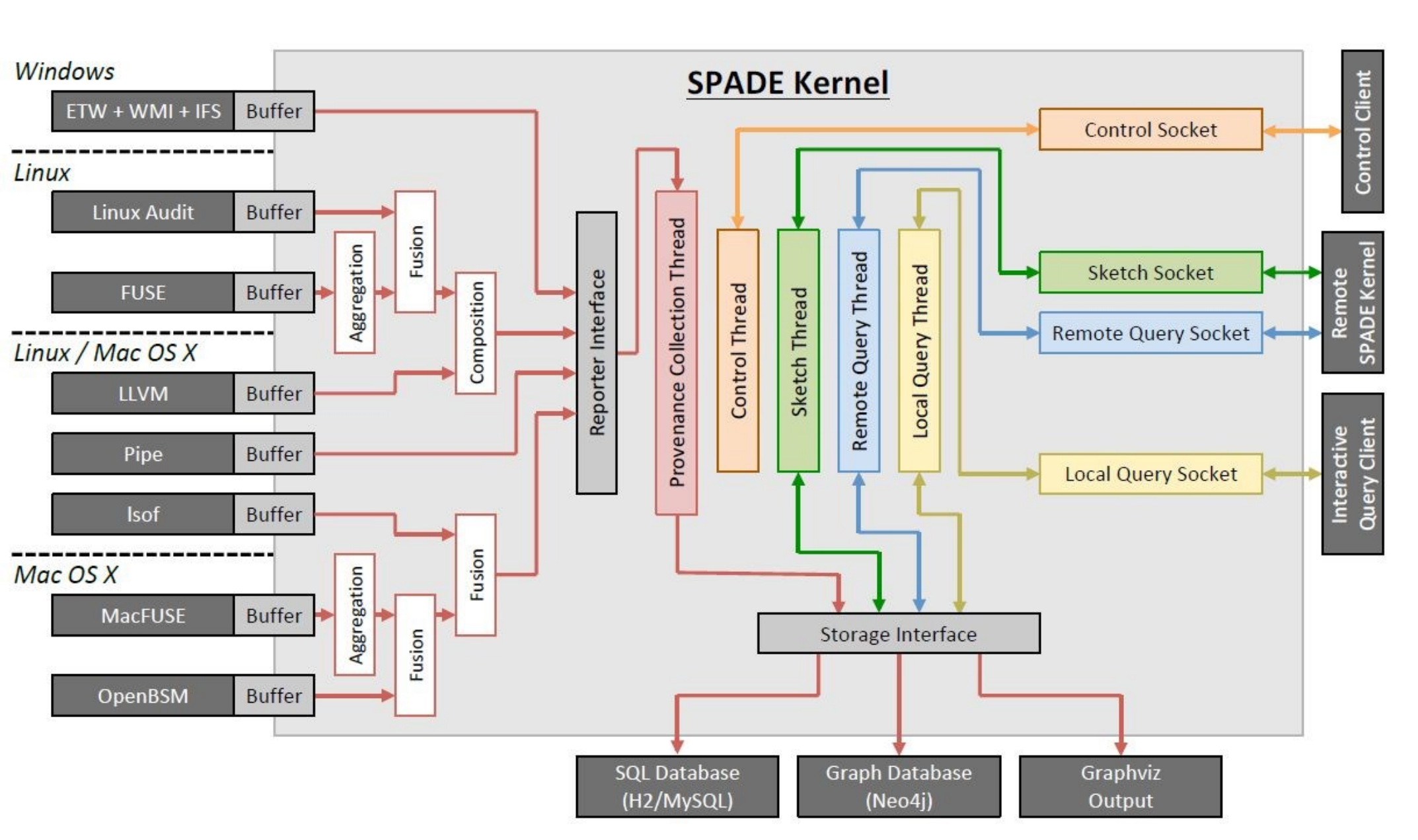
| Tool | Tracking Type / Level | PROV support | Storage support | For… |
|---|---|---|---|---|
| SPADE | Operation System | Yes | Yes | All |
| Sumatra | Version Control System | No | Yes | Python |
| YesWorkflow | Workflow Management | No | No | Python, R |
| PyPROV | API | Yes | No | Python |
| UniProv | Workflow Management | Yes | Yes | Unicore Workflows |

# SPADE Concept

- https://github.com/ashish-gehani/SPADE
- Motivation: filesystems report minimal information about history of stored data
- SPADE is a distributed service for collecting, storing, and querying provenance of data
- Cross-platform System-level capturing
- Support for Linux, Mac OS X, and Windows
- Uses the auditing functionality of each operating system

# SPADE Kernel

- *Reporters* collect provenance activities about domain of interest (I/O, network)
- *Filters* perform transformations on provenance events of interest
- *Storages* are introduced to record provenance in a new format
- written in Java
- multithreaded kernel runs as a daemon in the background
- at start it is deaf and blind to provenance reporting (to add reporters, filters, storages)
- control client for configuration (adding reporters, filters, etc.)


Windows
ETW + WMI + IFS
Buffer
Linux
Linux Audit
Buffer
FUSE
Buffer
Linux / Mac OS X
LLVM
Buffer
Pipe
Buffer
lsof
Buffer
Mac OS X
MacFUSE
Buffer
OpenBSM
Buffer
Aggregation
Fusion
Composition
Aggregation
Fusion
Fusion
SPADE Kernel
Reporter Interface
Provenance Collection Thread
Control Thread
Sketch Thread
Remote Query Thread
Local Query Thread
Control Socket
Sketch Socket
Remote Query Socket
Local Query Socket
Storage Interface
Control Client
Remote SPADE Kernel
Interactive Query Client
SQL Database (H2/MySQL)
Graph Database (Neo4j)
Graphviz Output

# Sumatra

- Aim of supporting reproducible research
- matured tool, started in 2013
- by tracking projects especially based on simulation and analysis
- can be thought of as an "automated electronic lab notebook"
- Sumatra consists of
  - a command line interface "smt"
  - a web interface
  - Python API
  - LaTeX package

# Sumatra – How does is work?

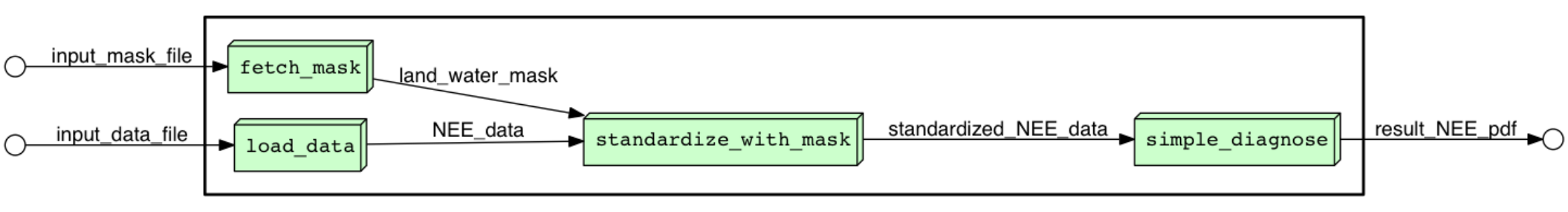
- Core library implemented as Python package
- interfaces on top of it (command-line, web, GUI interface)
  - to launch simulations with automated recording of provenance
  - managing projects (browse, view, delete)
  - to use sumatra directly in the code
- requires Django and django-tagging
- needs interaction with version control system (Subversion, Mercurial, Git, Bazaar)
- More in hands-on-session#1…

# YesWorkflow

- Aims to provide Provenance of scripted scientific workflows (Python, R)
- Without having to rewrite scripts and software
- Simply adds special YesWorkflow (YW) statements to existing scripts to declare
  - how data is used
  - how results produced step by step by a script
- Interpret YW statements e.g. in python scripts and produce graphical output
- Prospective provenance by parsing statements at compilation time

# YesWorkflow - Prototypes

```
# Example script at https://github.com/yesworkflow-org/yw-prototypes/blob/master/src/main/resources/example.py

# @BEGIN main
# @PARAM db_pth
# @PARAM fmodel
# @IN input_mask_file  @URI file:{db_pth}/land_water_mask/LandWaterMask_Global_CRUNCEP.nc
# @IN input_data_file  @URI file:{db_pth}/NEE_first_year.nc
# @OUT result_NEE_pdf  @URI file:result_NEE.pdf
```



# PROV Python Library (more in hands-on-session)

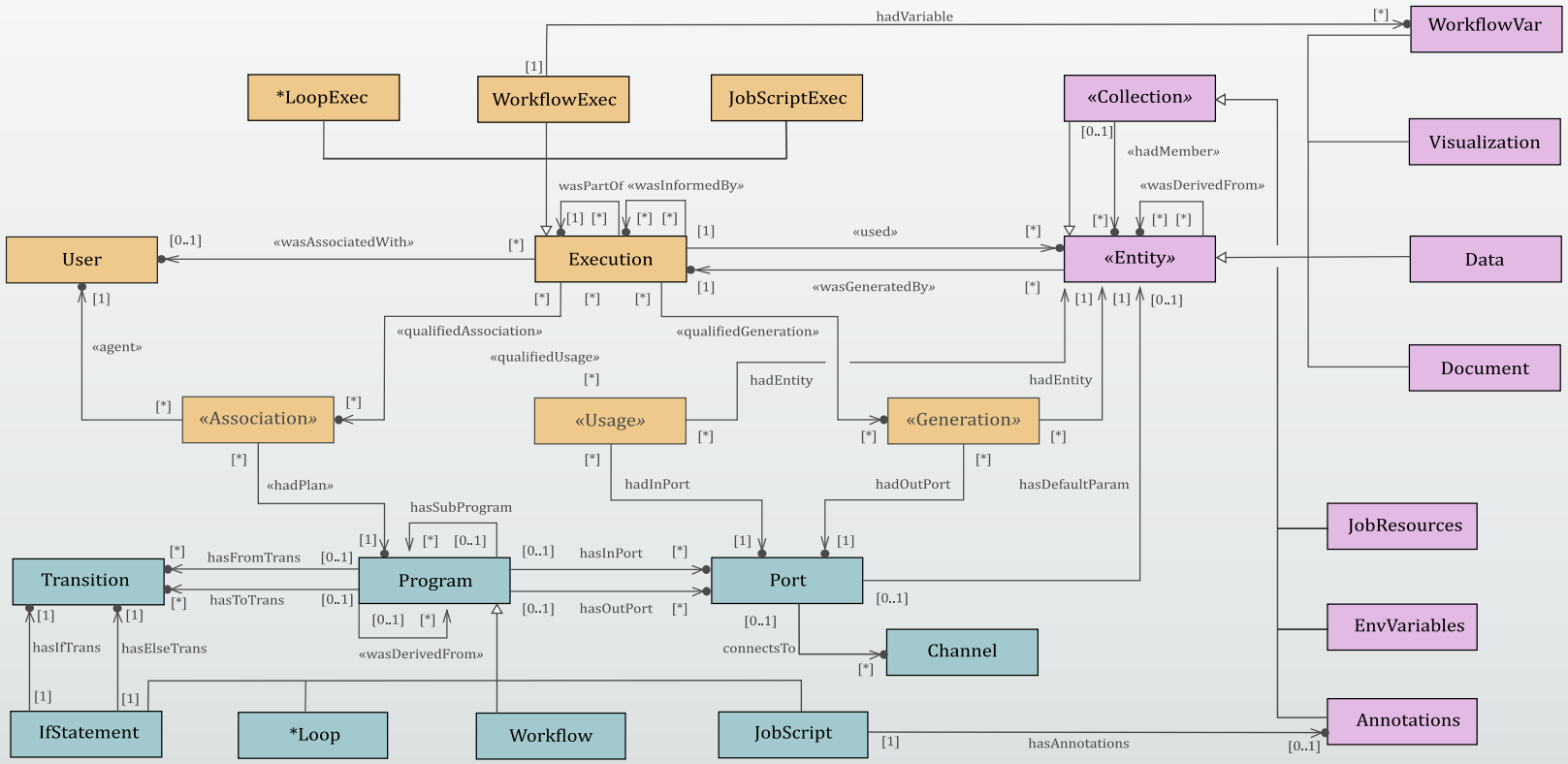
- https://github.com/trungdong/prov

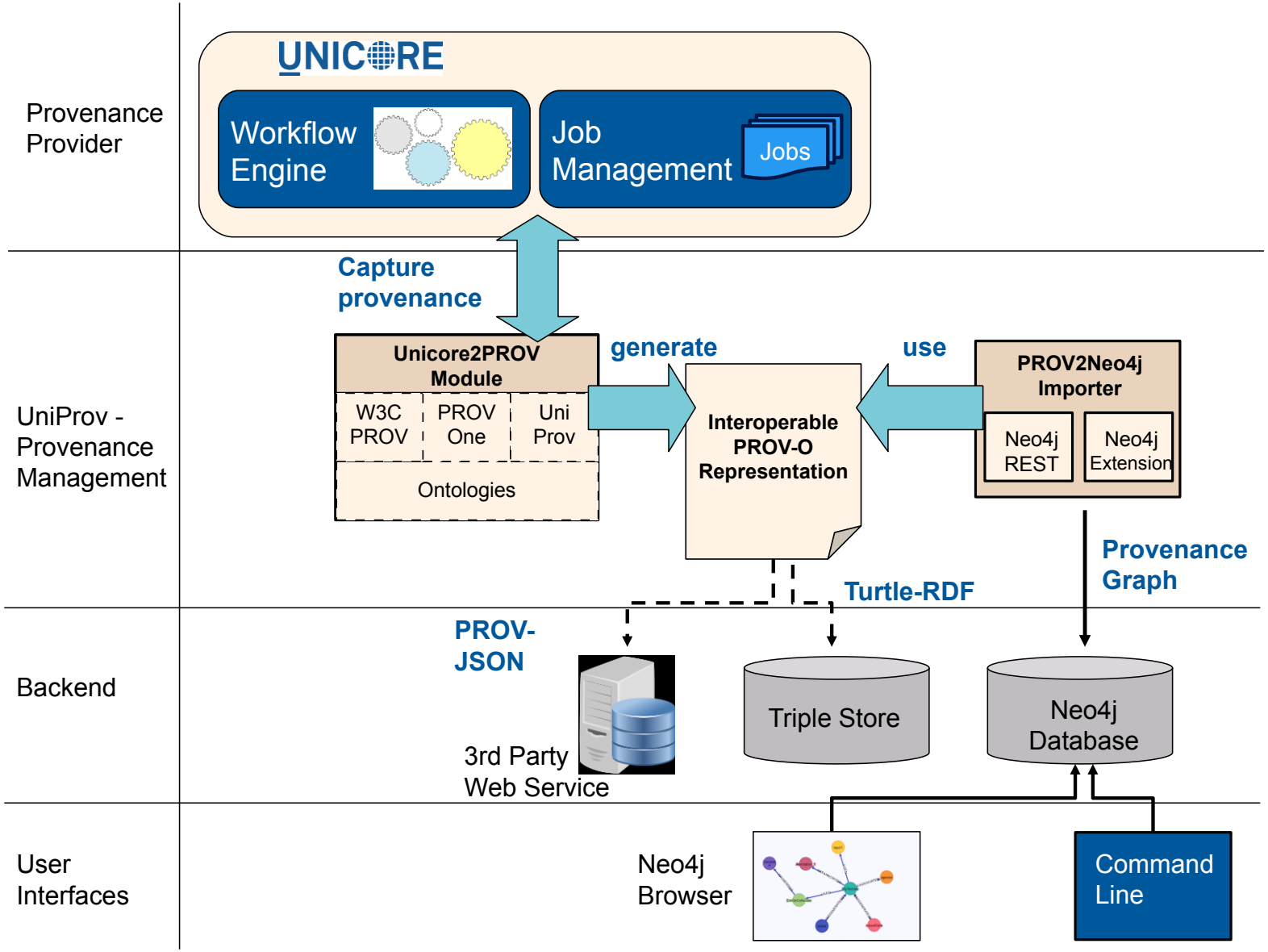
```
from prov.model import ProvDocument
# Create a new provenance document
d1 = ProvDocument()
# Entity: now:employment-article-v1.html
e1 = d1.entity('now:employment-article-v1.html')
# Agent: nowpeople:Bob
d1.agent('nowpeople:Bob')
# Attributing the article to the agent
d1.wasAttributedTo(e1, 'nowpeople:Bob')
d1.entity('govftp:oesm11st.zip',
                {'prov:label': 'employment-stats-2011',
                'prov:type': 'void:Dataset'})
d1.wasDerivedFrom('now:employment-article-v1.html',
                'govftp:oesm11st.zip')
# Adding an activity
d1.activity('is:writeArticle')
d1.used('is:writeArticle', 'govftp:oesm11st.zip')
d1.wasGeneratedBy('now:employment-article-v1.html', 'is:writeArticle')
```

# UniProv – Provenance for UNICORE Workflows

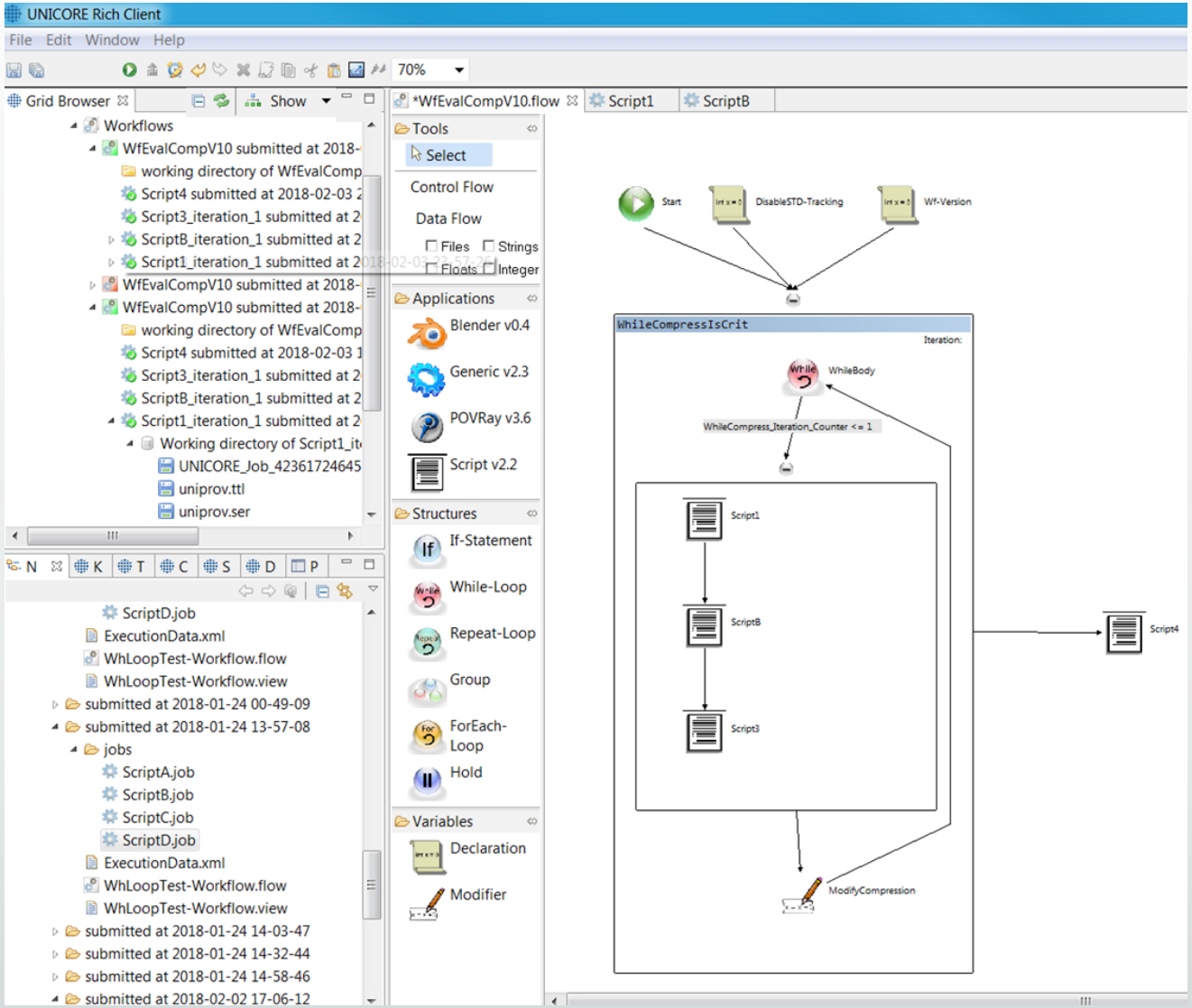
- Provenance Plugin for the UNICORE federation suite which allows tracking of job and workflow runs according to W3C-PROV
- Provenance Management Lifecycle – capturing, PROV modeling, storing (querying)
- Extracting and tracking Provenance Information from UNICORE job management and workflow engine service
- Processing the tracked Provenance data by generating an in-memory PROV model according to the ontology models (PROV-O, ProvONE, UniProv)
- Writing the Provenance output of the complete workflow in chosen syntax (e.g. RDF)
- Stores Provenance output in a Neo4j graph database

# UniProv Ontology extending ProvONE




Provenance Provider
UNICORE
Workflow Engine
Job Management
Jobs
Capture provenance
UniProv - Provenance Management
Unicore2PROV Module
W3C PROV
PROV One
Uni Prov
Ontologies
generate
Interoperable PROV-O Representation
use
PROV2Neo4j Importer
Neo4j REST
Neo4j Extension
Provenance Graph
Turtle-RDF
Backend
PROV-JSON
3rd Party Web Service
Triple Store
Neo4j Database
User Interfaces
Neo4j Browser
Command Line

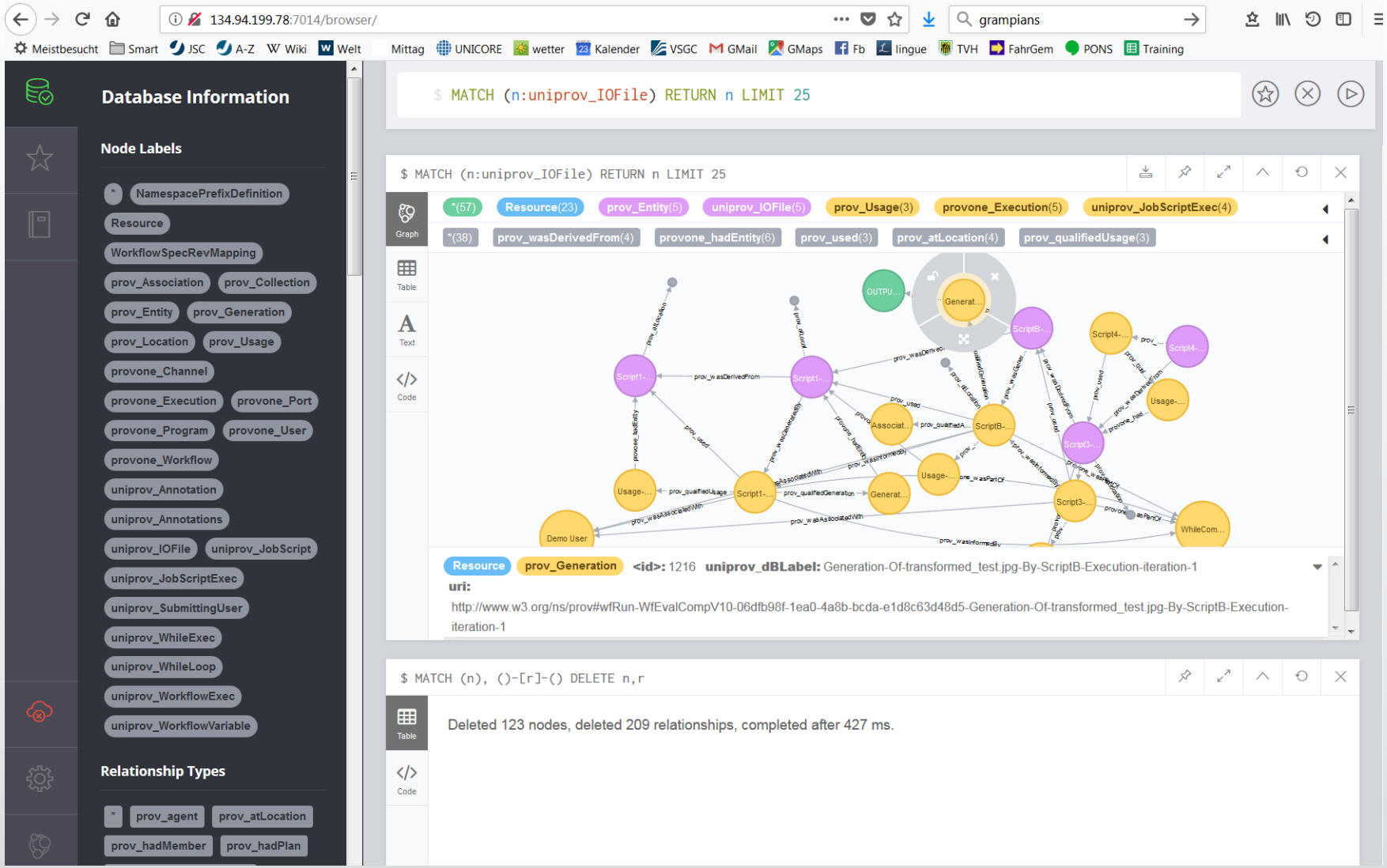
- UNICORE Rich Client
- Creating, submitting, and monitoring scientific workflows



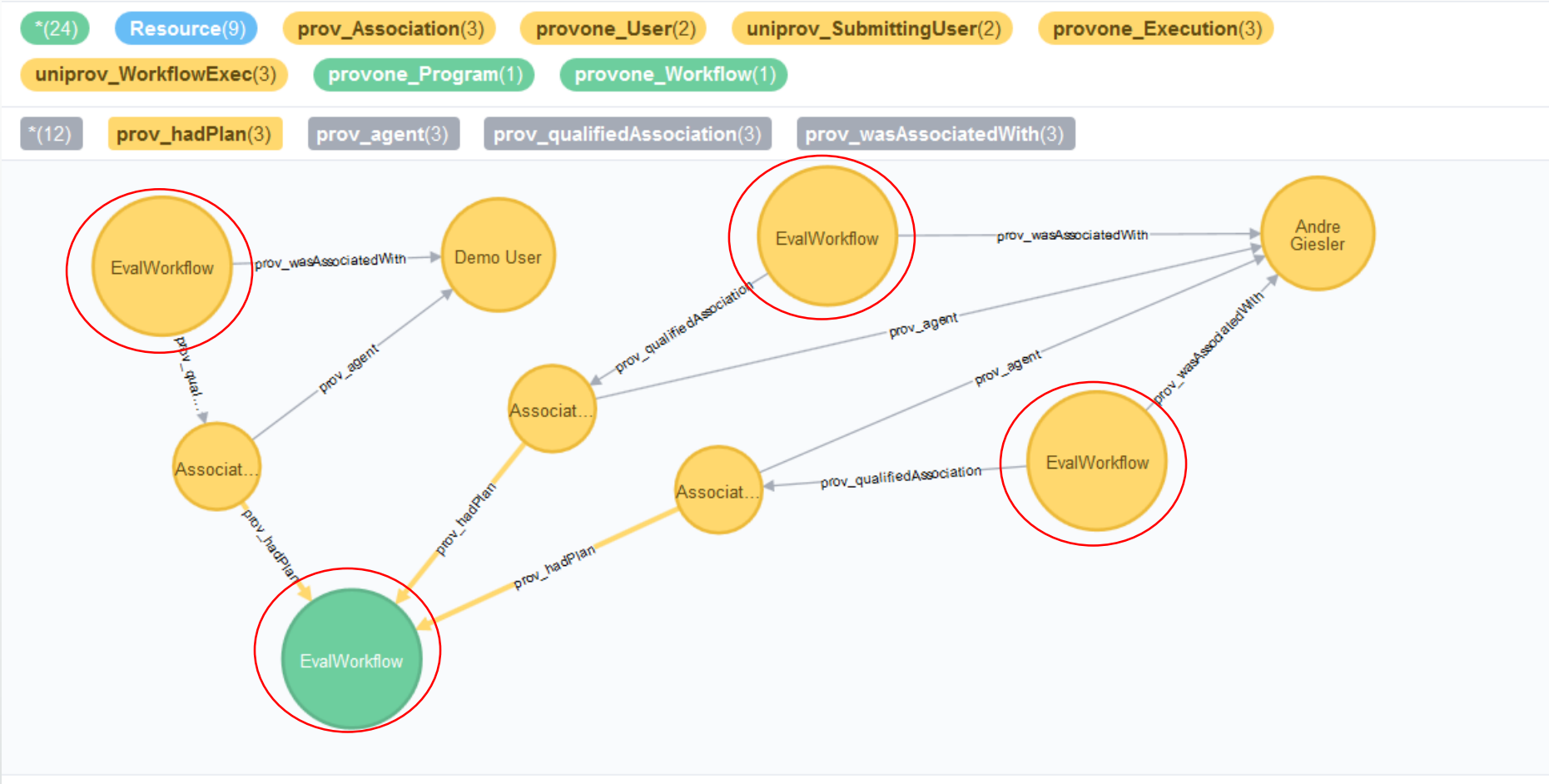
- Exemplary provenance output of a workflow in RDF
- Stored also in Neo4j database

```
@base          <http://purl.org/net/uniprov> .
@prefix uniprov: <http://purl.org/net/uniprov#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix provone: <http://purl.org/net/uniprov/provone#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix prov:  <http://www.w3.org/ns/prov#> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

uniprov:wfDesc-FreeNodeWorkflow-Script1-Version-1928282808
        a                       provone:Program , uniprov:JobScript ;
        uniprov:dBLabel         "Script1"^^xsd:string ;
        uniprov:hasToTrans      uniprov:wfDesc-FreeNodeWorkflow-Transition-Script1-to-WhileActivity1-Version-1928282808 ;
        uniprov:jobInterpreter  "/bin/bash"^^xsd:string ;
        prov:value              "ll\n"^^xsd:string .

uniprov:wfRun-FreeNodeWorkflow-8c9d55e3-540a-476f-8fe2-8a1e8c39b5f3-Script3-Execution
        a                         uniprov:JobScriptExec , provone:Execution ;
        uniprov:batchSystemId     "2156978374729"^^xsd:string ;
        uniprov:dBLabel           "Script3-Execution"^^xsd:string ;
        uniprov:dataMoveExecTime  "7.60 sec."^^xsd:string ;
        uniprov:exitCode          "0"^^xsd:string ;
        uniprov:jobId             "908525da-52f5-4fa4-967c-146cb82cee2e"^^xsd:string ;
        uniprov:jobWorkingDir     "/usr/local/unicore/unicore-servers-7.9.0/FILESPACE/908525da-52f5-4fa4-967c-146cb82cee2e/"^^xsd:string ;
        uniprov:machineInfo       "localhost"^^xsd:string ;
        uniprov:stageInExecTime   "1.18 sec."^^xsd:string ;
        uniprov:stageOutExecTime  "1.47 sec."^^xsd:string ;
        uniprov:status            "SUCCESSFUL [Success.]"^^xsd:string ;
        uniprov:totalExecTime     "34.92 sec."^^xsd:string ;
        provone:wasPartOf         uniprov:wfRun-FreeNodeWorkflow-8c9d55e3-540a-476f-8fe2-8a1e8c39b5f3 ;
        prov:atLocation           <file:///usr/local/unicore/unicore-servers-7.9.0/FILESPACE/908525da-52f5-4fa4-967c-146cb82cee2e/input> ;
        prov:endedAtTime          "2018-02-02T13:33:28.368Z"^^xsd:dateTime ;
        prov:qualifiedAssociation prov:wfRun-FreeNodeWorkflow-8c9d55e3-540a-476f-8fe2-8a1e8c39b5f3-Association-Of-Script3-Execution ;
        prov:startedAtTime        "2018-02-02T13:33:53.452Z"^^xsd:dateTime ;
```

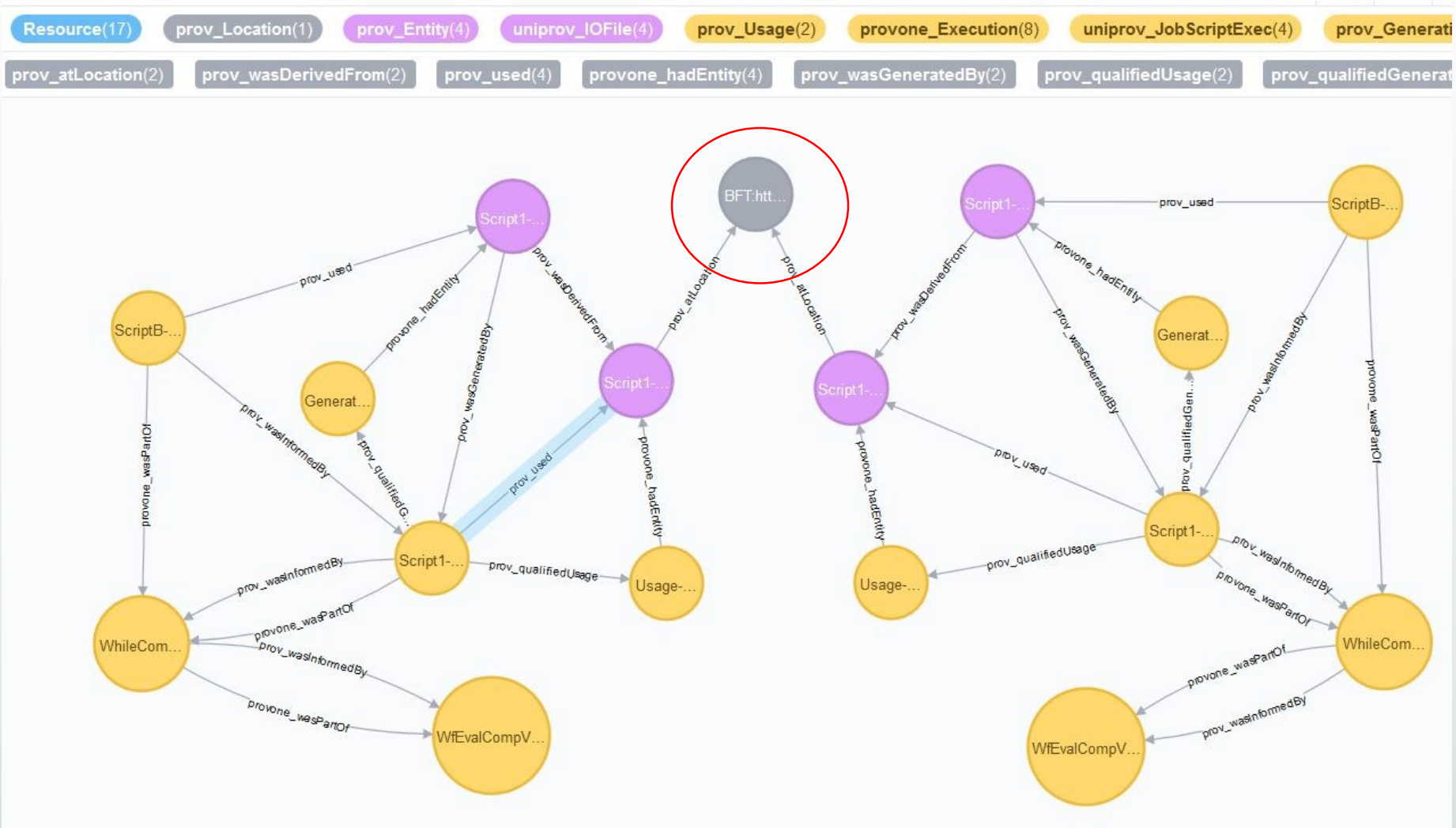
- Query provenance of submitted workflows with Neo4j browser



- Example#1: Find all executions of a specific workflow description



- Example#2: Find all workflow runs where a specific input file has been used



# Key Messages and Summary

- Recording Provenance is important
  - to understand where data came from
- Use a PROV standard if possible
  - A standard allows interoperability and comparison
- Storing Provenance in a (graph) databases, allows easy querying
- There is no magic Provenance Software doing all the work for you, but some tools that can provide some good starting points
- Recording Provenance often is a one-time effort
  - Develop a provenance approach, add capturing to applications, optionally include PROV mapping, store provenance data, and query data when needed
  - Once the provenance management works, track your tasks automatically

# Part 2 – Hands-on Session

- Exercise#1 – Sumatra
  - Tracking provenance of simulations applied with a python script
- Exercise#2 – PROV
  - Model an exemplary PROV document with PyPROV
- All Hands-on material can be found at sciebo: https://fz-juelich.sciebo.de/s/g85964rTTcCuC2I

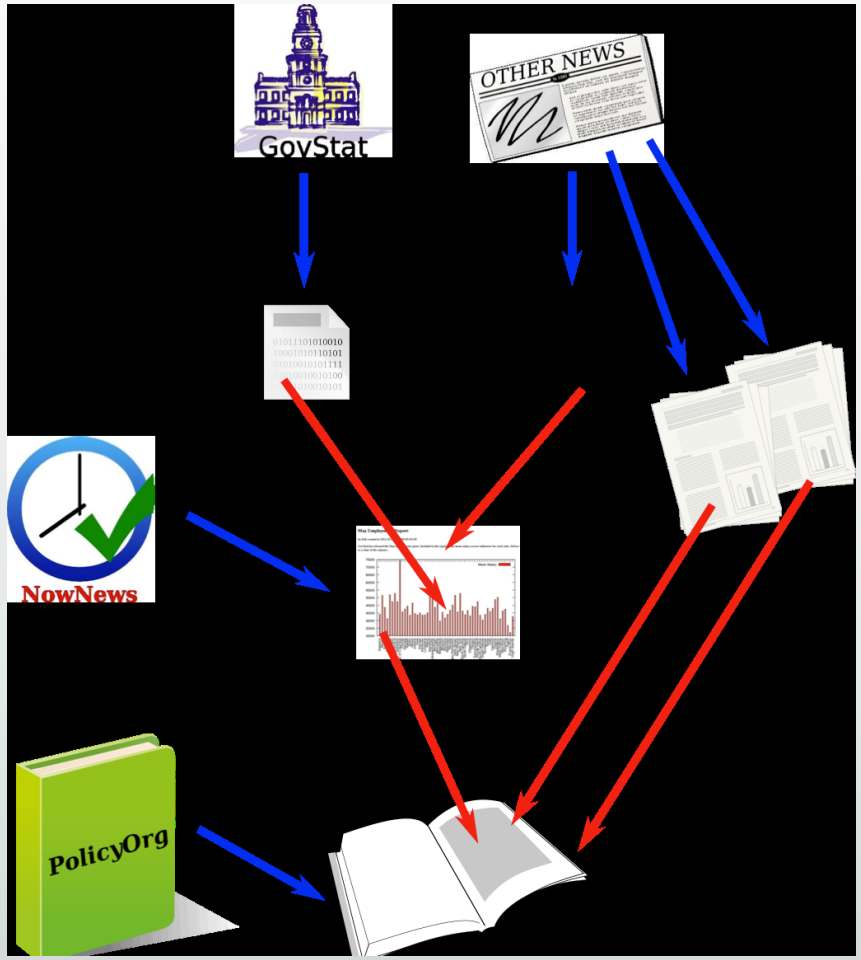
# Hands-on Session #1 – Sumatra Exercise

- **Step 1** – Download excercise1-sumtra.pdf from https://fz-juelich.sciebo.de/s/YoSf6hQPP3GygQ3 or https://b2drop-devel.zam.kfa-juelich.de/index.php/s/K76FaxGAedx5YJ5
- **Step 2, Option a** – Use your own laptop
- **Step 2, Option b** – If you are not sure if you apply the exercise on your own laptop, use our test virtual machine on the Juelich HAF cluster (will be deleted after workshop)
  - You've got a paper snippet containing a <user>, a <password>, and a <port>
  - Log in on zam10141 .zam.kfa-Juelich.de using your <user> and <password>
  - e.g. ssh <user>@ zam10141 .zam.kfa-Juelich.de
- **Step 3**
  - Read the downloaded PDF document and follow the instructions

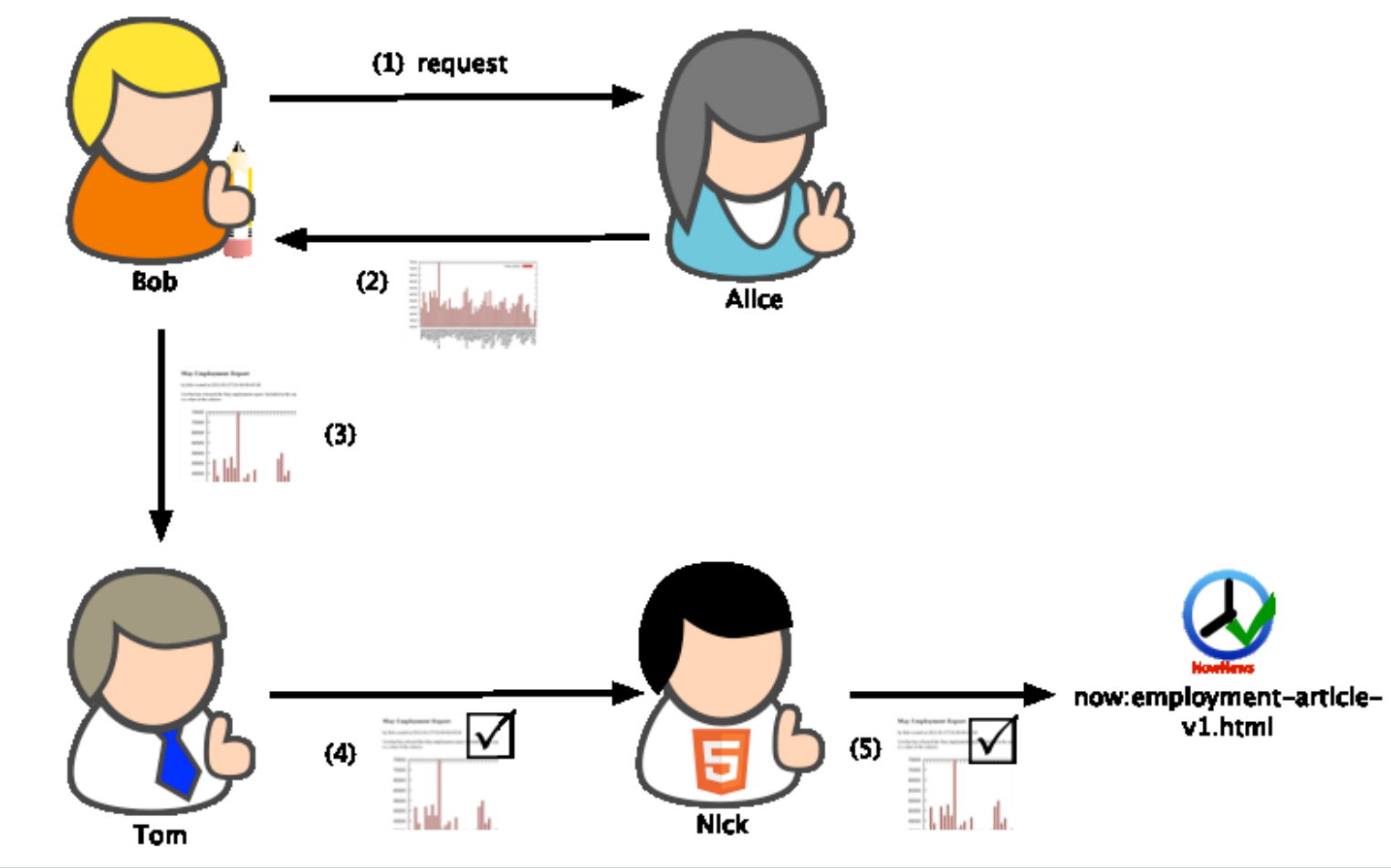
# Hands-on Session #2 – PROV Exercise - Background

Scenario

- NowNews publishes an article based on the latest employment data published by GovStat
- PolicyOrg compiles a report including NowNews article



# Hands-on Session #2 – PROV Exercise - Background




her

# Hands-on Session #2 – PROV Exercise

- **Step 1** – Download haf-workshop-provenance-excercise2-prov.pdf from https://fz-juelich.sciebo.de/s/PIZCYFPiJ7wnvzd
- **Step 2, Option a** – Use your own laptop
- **Step 2, Option b** – If you are not sure if you apply the exercise on your own laptop, use our test node on the Juelich HAF cluster (will be deleted after workshop)
  - You've got a paper snippet containing an <user>, a <Password>, and two <Ports>
  - Log in on zam10141 .zam.kfa-Juelich.de using your <Account> and <Password>
  - e.g. ssh <user>@ zam10141 .zam.kfa-Juelich.de
- **Step 3**
  - Read the downloaded PDF document and follow the instructions
  - Jupyter Notebook: https://fz-juelich.sciebo.de/s/gWexa5NUGDfMn6q/download