PROV-man: A PROV-compliant toolkit for provenance management

Ammar Benabdelkader, Antoine van Kampen, Silvia D Olabarriaga

Discoveries in modern science can take years and involve the contribution of large amounts of data, many people and various tools. Although good scientific practice dictates that findings should be reproducible, in practice there are very few automated tools that actually support traceability of the scientific method employed, in particular when various experimental environments are involved at different research phases. Data provenance tracking approaches can play a major role in addressing many of these challenges. These approaches propose ways to capture, manage, and use of provenance information to support the traceability of the scientific methods in heterogeneous environments. PROV is a W3C standard that provides a comprensive model for data and semantics representation with common vocabularies and rich concepts to describe provenance. Nevertheless, it is difficult for domain scientists to easily understand and adopt all the richeness provided by PROV. In this paper we describe the design and implementation of the provenance manager *PROV-man*, a PROV-compliant framework that facilitates the tasks of scientists in integrating provenance capabilities into their data analysis tools. *PROV-man* provides functionalities to create and manipulate provenance data in a consistent manner and ensures its permanent storage. It also provides a set of interfaces to serialize and export provenance data into various data formats, serving interoperability. The open architecture of *PROV-man*, consisting of an API and a configurable database, allows for its easy deployment within existing and newly developed software tools. The paper presents examples illustrating the usage of *PROV-man*. The first example illustrates how to create and manipulate provenance data of an online newspaper article using *PROV-man*. The second example demonstrates and evaluates the *PROV-man* implementation in a more complex case for collection of provenance data about biomedical data analysis activities that are carried out using a distributed computing infrastructure.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1102v1 | CC-BY 4.0 Open Access | rec: 20 May 2015, publ: 20 May 2015

PROV-man: A PROV-compliant toolkit for provenance management

A. Benabdelkader, A.H.C. van Kampen and S. D. Olabarriaga
Department of Clinical Epidemiology, Biostatistics and Bioinformatics
Academic Medical Center, University of Amsterdam, The Netherlands
e-mail: ammar@sharp-sys.nl, {a.h.vankampen, s.d.olabarriaga}@amc.uva.nl

Abstract

Discoveries in modern science can take years and involve the contribution of large amounts of data, many people and various tools. Although good scientific practice dictates that findings should be reproducible, in practice there are very few automated tools that actually support traceability of the scientific method employed, in particular when various experimental environments are involved at different research phases. Data provenance tracking approaches can play a major role in addressing many of these challenges. These approaches propose ways to capture, manage, and use of provenance information to support the traceability of the scientific methods in heterogeneous environments. PROV is a W3C standard that provides a comprensive model for data and semantics representation with common vocabularies and rich concepts to describe provenance. Nevertheless, it is difficult for domain scientists to easily understand and adopt all the richeness provided by PROV. In this paper we describe the design and implementation of the provenance manager *PROV-man*, a PROV-compliant framework that facilitates the tasks of scientists in integrating provenance capabilities into their data analysis tools. *PROV-man* provides functionalities to create and manipulate provenance data in a consistent manner and ensures its permanent storage. It also provides a set of interfaces to serialize and export provenance data into various data formats, serving interoperability. The open architecture of *PROV-man*, consisting of an API and a configurable database, allows for its easy deployment within existing and newly developed software tools. The paper presents examples illustrating the usage of *PROV-man*. The first example illustrates how to create and manipulate provenance data of an online newspaper article using *PROV-man*. The second example demonstrates and evaluates the *PROV-man* implementation in a more complex case for collection of provenance data about biomedical data analysis activities that are carried out using a distributed computing infrastructure.

Keywords

Provenance, OPM, PROV, e-science, database design, ER Modeling, RDBMS, open architecture, ORM, Java, Hibernate, Workflow management system.

1 Introduction

Many research laboratories nowadays use (new) technologies for large-scale data acquisition and distributed infrastructures for large-scale and collaborative data analysis. Research can take many years and involve a large number of people, data and tools. In such complex environment, proper methodologies need to be adopted by the scientists to carry out large endeavors in a way to guarantee that all the steps have been correctly performed and that they can be traced back to facilitate reproducibility of scientific results. The proliferation of large data sets and the increasing complexity of the scientific environment pose severe challenges for achieving this in practice.

Data provenance mechanisms provide ways to capture, manage, and use provenance information in heterogeneous environments [1]. They refer to the capability of determining the origin and history, or lineage, of a certain piece of data [2]. Therefore, data provenance plays a major role in addressing the emerging challenges in today's and future scientific environments. Additionally, the importance of data provenance is rapidly increasing in a connected digital world where open sources of data are becoming available for everyone [3].

In recent years, scientists and researchers from different application domains have increased their efforts in recording and exploiting data provenance facilities. The motivation for introducing mechanisms to manage data provenance in scientific experiments is two-fold. First, data provenance documents the data generation and analysis process by including how data and results were generated, and therefore it provides means to establish credibility and trust in scientific findings. Secondly, it provides useful means for the scientists to better understand the way they perform their experiments and to trace, reproduce and explain the data analysis process.

Provenance capture was and still is a crucial component in many developed software tools and applications [4] [5]. Most of these implement provenance in a manner very specific to their application domain or using specific concepts and technologies. Since the emergence of provenance as a standard (OPM [6] in 2007 followed by PROV [7] in 2013), many efforts have attempted to provide implementations of these standards[8]. Nowadays, PROV is being adopted by a large communinity from the scientific domain, therefore the number of related implementations rapidily increased. However, because the PROV definition is very detailed and complex, most of these implementations cover only part of the complete recommendations, and each focuses on one specific scientific domain. The lack of generic provenance tool means consumming a lot of efforts from experts in the scientific domain, and presenting additional challenges when new updates are introduced to the PROV standard. An exception is the ProvStore [9] and PROV-WF [10], which provide, respectively, a web service to manipulate provenance documents and a runtime provenance that can be queried even during the workflow execution. More clarifications about these development are given in section 2.3.

The main issue that remains unsolved for the scientist, even when using all these tools, is: *how can I instrument my scientific code to collect provenance data with less efforts and in a comprehensive and reliable manner?* Therefore, we felt the need to provide an implementation of PROV-compliant tools that facilitate the capture of provenance data with minimum effort by the developers of scientific applications and services.

In this paper we describe the design and implementation of a generic framework that is compliant with the provenance standard PROV, following the latest specification published by the provenance W3C community [7]. The implemented provenance management framework (*PROV-man*) consists of a programming interface (API) and a configurable database that can be used to create and store provenance according to the PROV standard. *PROV-man* deploys permanent back-end storage and follows an open architecture approach, which facilitates its deployment with existing and newly developed software tools. Interoperability and optimization are also considered at both the back-end storage and the core implementation of *PROV-man*.

In this paper we first introduce the provenance concepts (section 2), discussing their evolution in the domain of scientific applications, and highlighting the main efforts implementing provenance before and after the release of PROV. Section 3 presents the Implementation details of *PROV-man*, covering the approach, the database model and the API. Section 4 demonstrates the usage and deployment of *PROV-man* framework for provenance data creation and collection on a distributed computing infrastructure. Section 6 raises the implementation challenges and discusses their solutions. Finally, section 6 presents concluding remarks.

2 Provenance: Past and Future

Provenance, as general term, originates from the French *provenir*, "to come from". It refers to the chronology of the ownership, custody or location of a historical object. The term was originally mostly used for works of art, for which a good provenance helps to confirm the date, status, artist, subject, and the past owners of a painting, and can increase its value. Currently, the term Provenance is used in similar ways in a wide range of fields, including archaeology, paleontology, archives, manuscripts, printed books, and e-science [11].

In this section we present in more details the evolution of provenance in the context of e-science. The underlying assumption is that scientific research is generally considered to be of good provenance when it is sufficiently documented to allow reproducibility and to facilitate the process of tracking scientific datasets through all transformations, analyses, and interpretations. In the remaining sections of this paper we refer to Provenance in e-science as *data provenance*.

2.1 Early Efforts

At an early stage (before 1990), provenance information was mainly captured using unstructured logs and temporary files stored on the local disks of the machines where the programs are executed [12]. Provenance information has also been captured as metadata in information management systems for various applications. For example, DICOM (Digital Imaging and Communication in Medicine [13]) is a standard used for medical images that contains detailed information about the origin of medical images. Other examples are Laboratory Information Management Systems (LIMS) [14] and Electronic Laboratory Notebooks (ELN) [15], which have been around since the 90's and provide annotation facilities for workflow metadata and data tracking for experimental data.

From 2000, the use of data provenance terms for describing the history and lineage of data has become more prominent in scientific computing systems [12], [16]. In 2005, Yogesh [4] and Bose [5] published surveys and comparisons of the different projects and systems with mechanisms to manage data provenance. These projects cover different applications and disciplines such as Earth sciences [17], finances [18], e-science [19], curated databases [20], grid computing [3], and other projects such as Chimera [21], the Collaboratory for Multi-Scale Chemical Science (CMCS) [1], and Trio [2].

In the domain of e-science, the scientific workflow management systems (WfMS) developers were among the first interested in using and deploying provenance management. This is due to the step-wise design approach used for composing and executing workflows, which enables the capture of data provenance automatically and at fine granularity [22][23]. Examples of WfMS with provenance capabilities include Pegasus [24], Kepler [25], and Taverna [26]. Typically, each of the systems used its custom terminology for defining and capturing data provenance.

Around 2006, consensus about provenance concepts and terminology starts to emerge, and community efforts towards standardization become feasible as described below.

2.2 OPM: The Open Provenance Model

As a result of increasing interest in data provenance, in 2006 the International Provenance and Annotation Workshop (IPAW'06) [27] was organized. It involved around 50 participants, interested in the issues of data provenance, process documentation, data derivation, and data annotation. During the IPAW'06 workshop a consensus began to emerge on provenance standardization, hence a series of Provenance Challenges took place [28, 29]. As a result of this community effort, the *Open Provenance Model* OPM v1.00 was released in December 2007 [30]. The first OPM workshop, held in June 2008, involved around 20 participants who discussed issues related the OPM specification. This initiative led to a revised specification, referred to as OPM v1.01 [31].

OPM is based on three entities (*Artifacts*, *Processes*, and *Agents*) that are linked using *causal relationships*, representing their dependency (e.g. *used*, *wasGeneratedBy*, *wasControlledBy*, etc.).

OPM defines structures for representing the provenance information as a graph with nodes and edges, and also specifies inference queries. The original intent of OPM has been to define a data model that is open not only from an interoperability viewpoint, but also with respect to the community of its contributors, reviewers and users.

Since the release of OPM, various systems have been developed which implement OPM recommendations, or export provenance data using this model. These systems can be classified into two categories:

1) Specific systems with OPM import/export capabilities (e.g. Kepler/pPOD [32], Taverna Provenance [33], Karma [34], VisTrails [35], and Swift [36]).
2) Generic OPM-compliant frameworks to manage provenance data (e.g. PLIER [37], *e-BioFlow* [38]*, Karma [39], OPMProv [40],* Trident workbench [41], and SPADE [42]).

Many of these efforts shared a positive experience in using and deploying the OPM standard. In our PLIER implementation [37], briefly described in section 3, we shared the similar positive experience, although we outlined minor difficulties faced when implementing OPM or when making use of provenance data. Some of the outlined difficulties were: (1) the ambiguity of some terms and their usage (e.g. account, profile, and annotations), and (2) the improper design of some concepts (e.g. Time, Properties, and Relations). As a result of these experiences, OPM has been revised and improved since its release in 2007 by means of dedicated workshops, challenge series and community discussions.

2.3 PROV: the new release of a Provenance Standard

A major revision to OPM has been published in April 2013 as a W3C standard, under the name of PROV [7]. In a nutshell, PROV defines three core data types (*Entity*, *Activity*, and *Agent*); and *Relations* between these data types. *Attributes* can be defined for data and relations, and a *Document* aggregates them all.

PROV addresses most of the difficulties faced in OPM and provides a family of documents defining various aspects that are necessary to better achieve the vision of interoperability of provenance information in heterogeneous environments. PROV is conceived from a data modeling point of view and takes into account existing technologies in the field of information representation and data sharing. As such, it provides a set of classes, properties, and restrictions to model provenance information using semantic web technologies such as OWL2 ontologies, XML, and Dublin Core terms.

Figure 1 illustrates the organization of PROV components and the dependency between them. *PROV-DM* is the core conceptual Data Model that defines a common vocabulary and concepts used to describe provenance, to which a set of constraints apply as defined by *PROV-CONSTRAINTS* [7]. Other documents in the PROV family include the PROV OWL2 ontology to define the mapping of the PROV data model to RDF (*PROV-O*); an XML schema for the PROV data model (*PROV-XML*); a mapping between Dublin Core and PROV-O (*PROV-DC*); a declarative specification in terms of first-order logic of the PROV data model (*PROV-SEM*); how to use Web-based mechanisms to locate and retrieve provenance information (*PROV-AQ*); constructs for expressing the provenance of dictionary style data structures (*PROV-DICTIONARY*); extensions to PROV to enable linking provenance information across bundles of provenance descriptions (*PROV-LINKS*); and a human-readable notation for the provenance model (*PROV-N*).

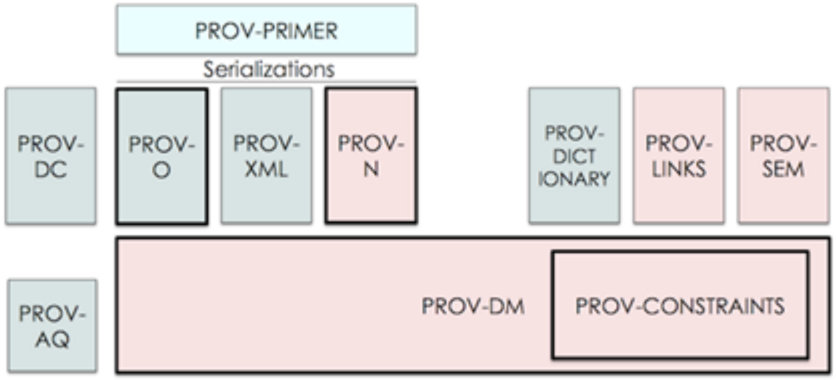

Figure 1: Organization of PROV according to [7] showing the core conceptual data model (PROV-DM), the family of documents it provides, and their dependencies. Bold bordered boxes denote W3C Recommendations, and regular bordered boxes denote Working Group Notes. The colors classify the audience for each document, namely: Users, Developers, and Advanced. Source: [7]

The major improvements introduced in PROV, particularly the PROV family of documents, have advanced the provenance standard to a level that attracted a large scientific community and increased the number of efforts in adapting to, and implementing PROV. The latest PROV implementation report, published in April 2013 [8], lists 66 implementations addressing PROV, classified into 5 types, namely: application, framework/API, service, vocabulary, and constraints validator. Most of the published implementations provide tools to convert and export between the different PROV families of documents, mainly to PROV-O, PROV-N, PROV-XML, and PROV-JSON, while others provide generic toolboxes and API frameworks for the management of provenance data. Nowadays, recent developments in the scientific and engineering areas are enhancing their software tools with provenance capabilities; examples include web semantics [43], data vizualization [44], decision making [45], scientific documentation [46], security controls [47], workflow systems [48] and many others. The provenance data collection in these developements usually consumes a lot of time and efforts. An out-of-shelf tool to help the developers of these applications collect and format the provenance data according to the PROV standard would aveliate them from this error-prone task and save their time and effort to better focus on the scientific applications.

The tools that are most related to our work are presented in [9,10]. Huynh et al. [9] provide ProvStore: a web service to store, browse, visualize, share and manage provenance documents. ProvStore expects the user to have the data already collected in a given format and provides no means to collect the data. Flavio et al. describe in [10] RPOV-wf, a PROV-based database to provide runtime provenance that can be queried even during the workflow execution. The approach collects runtime provenance data from the various WfMS execution engines into the centric database.

To our knowledge, to date, none of these implementations provide a generic framework that is open enough to be incorporated and deployed into scientific software tools and systems to facilitate the capture of provenance in full-compliance with PROV.

3 PROV-man: Design and Implementation

This section presents the background of the design of *PROV-man*, which is the framework we developed to facilitate the creation, storage, management and access to provenance data according to the PROV standard recommendations. After presenting some background information, the approach adopted for the data model optimization and the framework implementation are described.

3.1 Background

We have been involved in the design and implementation of a provenance framework for both OPM and PROV. Our former implementation of provenance management was based on OPM and called Provenance Layer Infrastructure for e-Science Resources – PLIER [37]. It was conceived based on an optimal database schema to store provenance for scientific experiments that are performed using gri d workflow management systems. PLIER provides an API to record information about the steps of experiments, their order, and the cause-and-effect reflecting linkage of inputs to output results. Additionally, we enhanced PLIER with a set of tools to build, store, retrieve, share, and visualize workflow experiments. PLIER has been extensively used to collect and explore provenance for scientific experiments performed on a grid infrastructure, namely: (1) as an integrated component within the WS-VLAM workflow system [49,50], and (2) as a core component to automatically gather provenance data from existing grid workflow enactments services [51,52]. The results achieved by deploying PLIER for tracing and analyzing the results of experiments motivated us to proceed with the implementation of the provenance framework according to PROV.

OPM	PROV
Graph	Document
Artifact	Entity
Process	Activity
Causal Dependencies	Relations
Annotation & Property	Attributes
Account, Profile, OTime	N.A.

Table 1: Relation between OPM and PROV concepts

First, we conducted a study comparing PROV to OPM, based on the provenance specifications as defined for OPM Core Specification (v1.1) and the latest PROV documentation [7]. Table 1 illustrates the main OPM concepts with their counterparts in the PROV specification. In more details:

- The concepts *Graph*, *Artifact*, *Process*, and *Causal Dependency* have been renamed to *Document*, *Entity*, *Activity*, and *Relation*. These new terms are more suitable and representative in the domain of data management.
- The concepts *Annotation* and *Property* have been refactored and simplified to *Attributes*, which facilitates their use and deployment.
- The concepts *Account* and *Profile* are not present in PROV[1].

Other changes have been also introduced to the structure of the *Relation* and *Activity* concepts in PROV, which make their representations more descriptive (e.g. by adding *Start Time* and *End Time* for the Activity).

The main conclusion of our study is that the PROV modeling concepts are more appropriate than their OPM counterparts. Particularly, the relationships concepts in PROV are conceived with rich attributes, which provide comprehensive mechanisms to better describe the semantics of data.

[1] In our deployment of PLIER for collecting provenance data, we did not encounter effective usage for those concepts.

3.2 PROV-man: The Approach

The design and implementation of *PROV-man* follows the PROV recommendations and considers these main design requirements:

1) To provide permanent storage of provenance data,
2) To optimize the database model considering data representation and querying,
3) To implement functions to facilitate access to provenance data,
4) To support data sharing via a set of utility functions for data conversion to various standard formats,
5) To allow for easy deployment of the framework in various use cases.

The main components of the framework consist of a database implementing the PROV-DM concepts (section 3.3), and an API implementing the set of classes with methods and utility functions (interfaces) to create and manipulate provenance data represented according to this model (section 3.4).

3.3 PROV-man Optimized Data Model

Data provenance is described in PROV by the use and production of *Entities* by *Activities*, which may be influenced in various ways by *Agents*. *PROV-DM* is the core conceptual data model that defines a common vocabulary and concepts used to describe provenance. In brief, PROV-DM consists of:

a) *Core data types* (*Entity*, *Activity*, and *Agent*);
b) A set of *Relations* between the core data types as defined in PROV (16 in total);
c) A set of *Attributes* that can be defined for each of the core data types and Relations, describing their properties as key-value pairs; and
d) A *Document* grouping all the above.

Figure 2 illustrates a subset of the entity-relationship (ER) diagram of the PROV-DM core data types and their *Relations*. Note that the complete ER diagram would be too complex to display because it would include all optional *Attributes* that can be defined for the core data types and *Relations*.

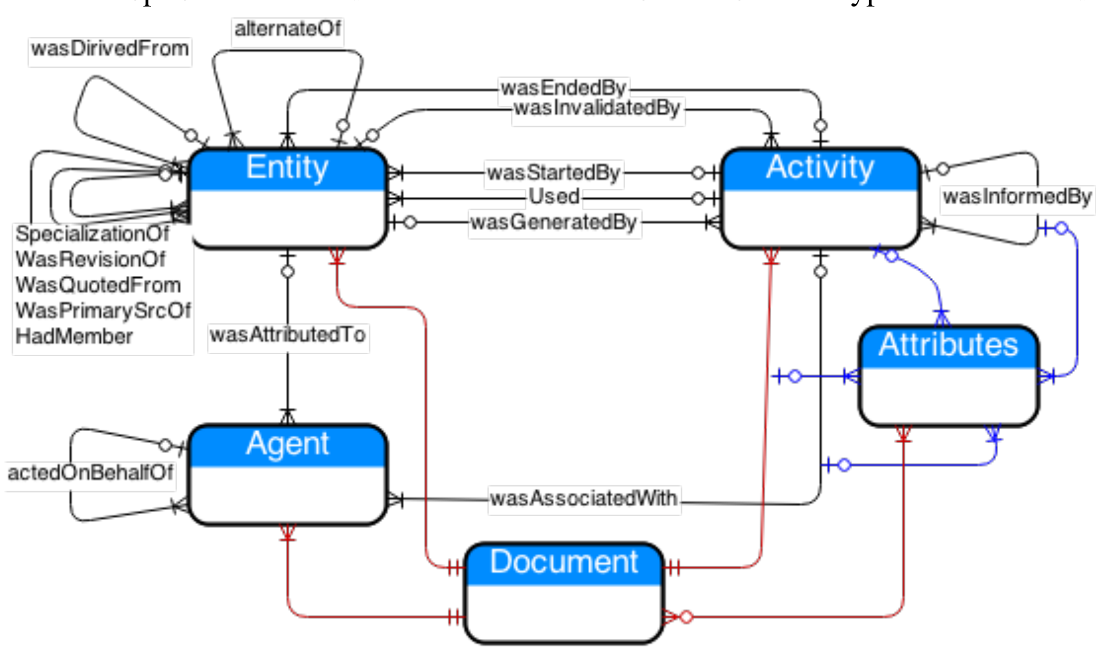

Figure 2: PROV-DM core data types with their prominent relationships. For readability reasons, only a subset of the relationships to the Attributes (highlighted in blue) are presented.

Relations in PROV-DM are always defined between the three core data types: *Entity*, *Activity*, and *Agent*. Their richness provides a strong mechanism to describe and express semantics of data. In addition, *Attributes* allow for further description of the core data types and their relationships. The strict implementation of this data model, without optimization, would however introduce difficulties for querying and maintaining the provenance data. For example, to retrieve the *Relations* for a given *Entity*, separate queries would be required for each of the 13 *Relations* defined for that *Entity*. Moreover, all the *Relations* have *Attributes*, for which separate tables would be also needed, thus making the data model even more complex. Thus, there is a need to optimize the data model to guarantee simplicity and high efficiency when querying the provenance data. The challenge here is to optimize the number of tables in PROV-DM, while preserving the full semantics and data richness of those relationships.

From a database design perspective, an optimization could be to model all the Relations using a single table. We demonstrate the optimization approach using the example in Figure 3, illustrating three of the 16 *Relations* defined in PROV-DM. As shown on this example, *Relations* are structurally similar to each other. For example, the relationships *used* and *wasGeneratedBy* are almost the same, except for the roles of the cause and effect, which are reversed (*Entity* and *Activity*). In the *actedOnBehalfOf* relationship, both cause and effect point to objects of the same data type (*Agent)*, with an additional field *Activity* for which the delegation took place.

Definition of Relations

used(Identifier, *Activity*, *Entity*, Time, Attributes)
wasGeneratedBy(Identifier, *Entity*, *Activity*, Time, Attributes)
actedOnBehalfOf(Identifier; *Agent*, *Agent*, Activity, Attributes)

Examples of Relations creation

Entity (e1); Entity (e2); Activity (a1); Agent (ag1); Agent (ag2); // given
used ('r1', a1, e1, '23:09:2013 14:04', -); // activity a1 used entity e1 at '23:09:2013 14:04'
wasGeneratedBy ('r2', e2, a1, '24:09:2013', -); // entity e2 wasGeneratedBy activity a1 at '24:09:2013 10:04'
actedOnBehalfOf ('r3', ag2, ag1, a1, -); // agent ag2 actedOnBehalfOf agent ag1 for activity a1

Figure 3: Examples illustrating three *Relations* expressed using PROV-N notation

Therefore, we have chosen to model all PROV *Relations* using a single table:

***Relation** (Identifier, RelationType, Cause, Effect, Time, Activity, Usage, Generation, Entity, Attributes)*

Definition of Relations

Relation (Identifier, RelationType, Cause, Effect, Time, Activity, Usage, Generation, Entity, Attributes)

Examples of Relations creation

Entity (e1); Entity (e2); Activity (a1); Agent (ag1); Agent (ag2); //given
Relation('r1', "*Used*", a1, e1, '23:09:2013 14:04', -, -, -, -, -);
Relation('r2', "*wasGeneratedBy*", e2, a1, '24:09:2013 10:04', -, -, -, -, -);
Relation('r3', "*actedOnBehalfOf*", ag2, ag1, -, a1, -, -, -, -);

Figure 4: Example of Relations from Figure 3 after optimization, using a single relationship that specifies the RelationType.

The member *RelationType* plays the role of discriminator and ensures the preservation of the relationships semantics. Two keys (*Cause* and *Effect*) can point to a foreign key in one of the three other tables (*Entity*, *Activity* and *Agent*). *Time*, *Activity*, *Usage*, *Generation* and *Entity* are optional fields (see more details about these fields in [7]). Figure 4 illustrates how the class hierarchies of the three PROV-DM relationships in Figure 3 are modeled using this optimized model.

This optimization approach can be applied to all the sixteen PROV-DM Relations, thus reducing the number of relationships to a single table *Relation*. Consequently, the number of *Attributes* describing the properties of the *Relations* will be also reduced to a single table *RelationAttributes*.

Figure 5 depicts the *PROV-man* data model in which the PROV-DM *Relations* are re-arranged in a manner that reduces the model complexity and preserves PROV full semantics. A *Document* is made of a set of *Entities*, *Activities*, and *Agents*; *Relations* may be established between the three core data types; and each of the components can be further described using a set of *Attributes*.

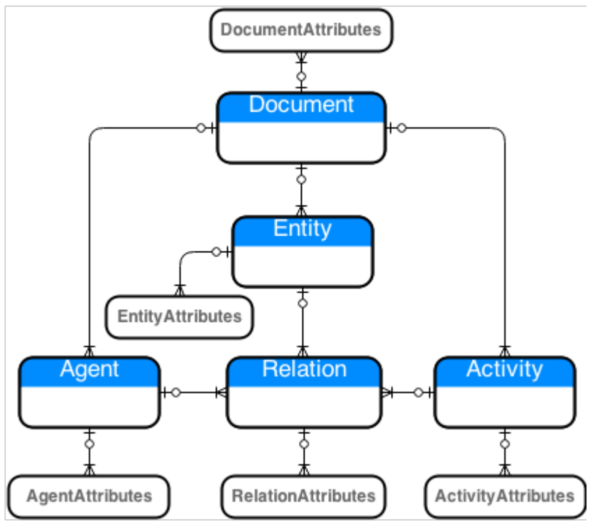

Figure 5: Optimized PROV-man data model.

In *PROV-man* we dedicate special attention to the optimization of the underlying database schema, so that it become simpler and more efficient for querying or storing provenance data, in case the scientist needs/prefers direct access to the database. Still, direct access to the database is only suggested for users with advanced database and PROV knowledge.

3.4 PROV-man API implementation

The *PROV-man* API provides an interface to create and manipulate provenance data according to the PROV specifications. It preserves the semantics and richness defined by PROV and makes the *PROV-man* data model transparent to the application developer. *PROV-man* software release and documentation in are available in [53]. Figure 6 depicts the open-architecture of the *PROV-man* framework, providing:

- A set of *classes* with methods to build and manipulate provenance data according to PROV specifications;
- A set of *interfaces* implementing utility functions for provenance sharing and interoperation.
- A back-end database that serves as a main repository for storing provenance data, reflecting the *PROV-man* data model presented in Figure 5; and

- Object-relational mapping (ORM) between the Java objects (classes) and the relational database.

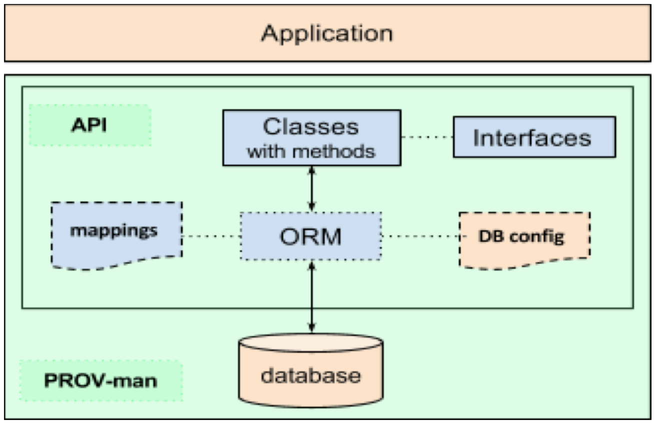

Figure 6: PROV-man architecture consisting of a database and an API. Components highlighted in brown denote the parts that can be controlled

The Java programming language has been selected to realize the implementation of the *PROV-man* framework. In addition, ORM technology was used to implement the mapping between the relational *PROV-man* data model and the Java object-oriented programing language. The choices and motivations for selecting the technologies to implement the *PROV-man* framework are the following:

- A relational DBMS is used as back-end storage, which allows for remote and distributed access, enforces data integrity, and serves as a distributed repository for provenance data. *PROV-man* deploys an XML-configuration file to specify the underlying database with connection and tuning parameters (e.g. database URL, user name and credentials, connection pool parameters, and cache level) .
- Java was selected for the implementation of the *PROV-man*, due to its portability, platform independency, and richness for modeling the provenance concepts and relationships. Provenance data is created and consolidated as Java objects and then stored into the relational *PROV-man* database.
- Hibernate [54] is used for the mapping between domain objects and relational database, which permits to select a different DBMS if needed. It provides a smooth mapping between the Java classes reflecting PROV-DM and the *PROV-man* optimized relational data model.

The *PROV-man* core API provides a set of 24 classes implementing the PROV-DM core data types, their relationships, and attributes. Figure 7 illustrates an example of methods implemented for the PROV-DM *Activity* class and Figure 8 illustrates methods for the PROV-DM *wasDerivedFrom* relation. Figure 8 also illustrates that the naming of methods and parameter types are enforced accordingly to the specification given by PROV-Constraints.

```
setId(String id) : void - Activity
setStartTime(Date starttime) : void - Activity
setEndTime(Date endtime) : void - Activity
setAttributes(List<ActivityAttributes> activityAttributes) : void - Activity
```

Figure 7: Methods implemented for *Activity*. Each method has parameters and returning value. Similarly, *get* methods exist to retrieve these values.

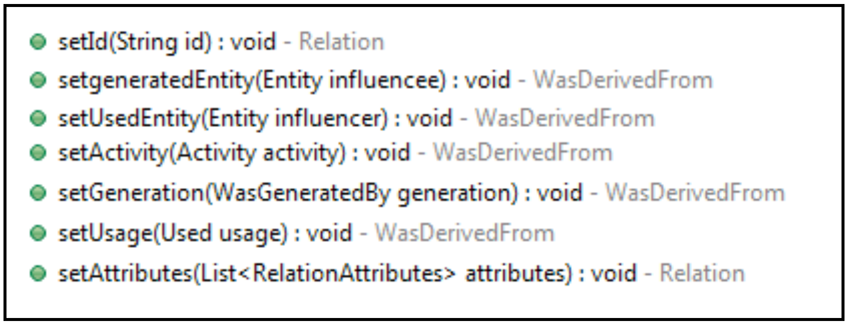

Figure 8: Methods implemented for *wasDerivedFrom*. Each method has parameters and returning value. The terms in grey indicate whether the method is generic for all *Relation* types or specific to *wasDerivedFrom*

To facilitate the creation of provenance data, *PROV-man* also provides a set of additional methods following a human readable notation. These methods are provided under *PROVmanFactory* and follow a syntax similar to PROV-N. Examples on the usage of the *PROV-man* methods and interfaces are illustrated in section 4.1.

Finally, a set of interfaces cover serialization into formats of the PROV family of documents and other formats:

- *toDB (document)*: maps the provenance *document* from its o-o representation to a relational model, using ORM concepts, and stores it into the *PROV-man* database;
- *toXML(document, filePath)*: serializes the provenance *document* to the corresponding XML representation, in compliance with the PROV XML schema;
- *toProvN(document, filePath)*: serializes the provenance *document* to the human-readable notation of PROV-N;
- *toOWL2(document, filePath)*: serializes the provenance *document* to the corresponding Web Ontology Language (OWL2-RL) representation;
- *toGraphviz(document, filePath)*: translates the provenance *document* to the Graphviz DOT format [55];
- *toGraph(document, format, filePath)*: generates a graphical representation of the provenance *document* , according to the specified format (e.g. png, jpg, gif, and pdf). This interface relies on the Graphviz software [55], which supports most of the graphical output formats.

These interfaces take a generic and basic serialization approach that can be useful for getting started; they are distributed as examples that possibly need to be customized for a particular application or usage scenario.

4 PROV-man Usage Examples

The open architecture of the *PROV-man* framework, illustrated in Figure 6, allows for its flexible integration into existing and newly developed software tools. The application layer can consist of existing software (e.g. workflow systems or some data analysis tool) that deploys and integrates *PROV-man* into its core implementation to store the fine-grained provenance details. *PROV-man* can be used to build provenance extraction tools, for example, to gather provenance data from logs or other information sources available for an application or system. *PROV-man* could be also deployed in scenarios where multiple provenance tools/applications share the same *PROV-man* database by using the same database configuration.

Below we present two usage examples: a simple case that illustrates the use of the set of methods and interfaces provided by *PROV-man* in a stand-alone program, and a more complex case, which demonstrates the deployment of *PROV-man* into a science gateway.

4.1 Simple Example: online newspaper article

Here we present and discuss the implementation of an online newspaper article described in the PROV-PRIMER [56]. The newspaper publishes an article with a chart about crime statistics based on existing data, with values composed (aggregated) by geographical regions. Different namespace prefixes are used to identify the source creating the data and to distinguish between identifiers with the same name used in these sources (e.g. **exb**, **exn**, **exc**, and **exg**). Figure 9 shows part of the Java code to create data provenance. The complete code and the implementation details of this example are available at the *PROV-man* release page [53].

```
01: Document document = new Document();
02: Entity e1 = new Entity();    e1.setId("exg:DataSet1");
03: Entity e2 = new Entity();    e2.setId("exc:RegionList1");
04: document.getEntities().add(e1);   document.getEntities().add(e2);
05: Entity e3 . . .
06: Activity act = new Activity();  act.setId("exc:Compose1");
07: document.getActivities().add(act1); . . .
08: Activity act3 = new Activity(); . . .
09: ActivityAttributes Attr = new ActivityAttributes();
10: Attr.setId("Status");  Attr.setValue("Planned");
11: act.getAttributes().add(Attr);
12: document.getActivities().add(act3);
13: Agent agent = new Agent(); agent.setId("exc:derek");
14: document.getAgents.add(agent);
15: Agent agent2 = new Agent(); . . . . .
16: WasAssociatedWith waw = new WasAssociatedWith();
17: waw.setId("waw"); waw.setActivity(act2);
18: waw.setAgent(agent); waw.setPlan(e1);
19: document.getRelations().add(waw);
20: ActedOnBehalfOf abo = PROVmanFactory.ActedOnBehalfOf("abo",agent,agent2);
21: WasAttributedTo wat = PROVmanFactory.WasAttributedTo("wat", e4, agent);
22: document.getRelations().add(abo);   document.getRelations().add(wat);
23: Used used = PROVmanFactory.Used("used", act, e1,"prov:role", "exc:dataToCompose");
24: document.getRelations().add(used);     Used used2 . . .
25: WasGeneratedBy wgb= PROVmanFactory.WasGeneratedBy("wgb",e3,act,"prov:Role","exc:composedData");
26: document.getRelations().add(wgb);
27: WasDerivedFrom wdf = PROVmanFactory.WasDerivedFrom("wdf",e4,e3, "prov:type", "prov:Revision");
28: document.getRelations().add(wdf);
29: PROVman.toDB(document);
30: PROVman.toXML(document, “/home/PROVman/doc/xml”);
31: PROVman.toGraphviz(document, “/home/PROVman/doc/dot”);
32: PROVman.toGraph(document, “png” , “/home/PROVman/doc/png”);
```

create provenance data objects (lines 01–03)

establish the link between data objects using relationships (lines 09–12)

*Use of PROVman*Factory to simplify the creation of provenance data using syntax similar to PROV-N (lines 20–28)

Figure 9: Java sample code illustrating the use of PROV-man for creating and manipulating provenance data.

Figure 9 also illustrates calls to the *PROV-man* interfaces for interoperability and data sharing (lines 29-32). The corresponding data provenance graph generated by the *toGraph()* function for the on-line newspaper article is depicted in Figure 10.

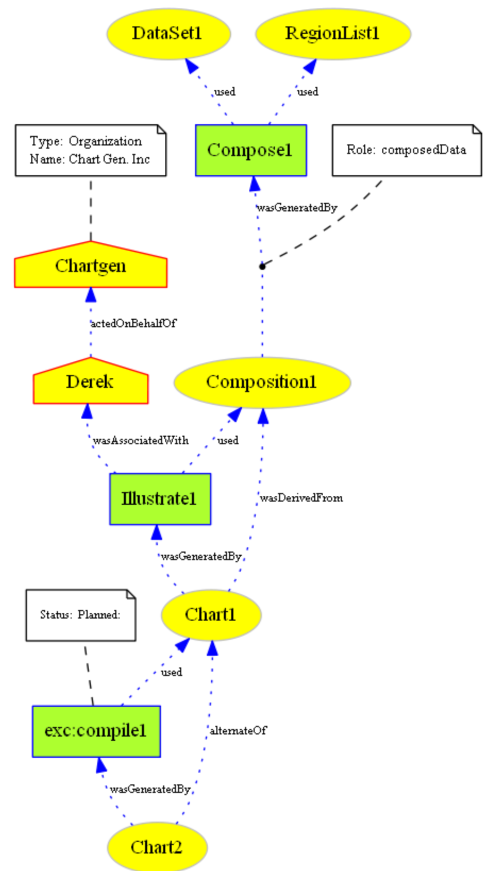

Figure 10: Data provenance graph corresponding to the online newspaper article generated by toGraph() function.

4.2 Provenance of a science gateway

Here we demonstrate the deployment of *PROV-man* within an existing system, namely the AMC Neuroscience Gateway (NSG) [57]. This section briefly introduce the approach used to collect provenance using *PROV-man*. More details about the usage of the collected provenance data and the potential for their exploration can be found on [58].

The Neuroscience Gateway (NSG) is deployed at the Academic Medical Center (AMC) of the University of Amsterdam (UvA), The Netherlands. Its design is based on the WS-PGRADE/gUSE [59] scientific workflow management portal and framework, which supports various distributed computing infrastructures (DCIs). The gateway simplifies the usage of the Dutch e-Science Grid [60] for biomedical researchers by providing services such as community grid certificate and automatic file transport between the data servers and the grid resources. Workflows implemented using the WS-PGRADE/gUSE framework are the core of this platform. The workflows implement the data analysis tools for different applications (e.g. neuroscience and DNA sequencing). The users of the Neuroscience gateway are biomedical researches who perform data analysis tasks (coined *experiments*) by running these workflows on their data sets. Finally, the workflows are executed on the grid infrastructure by the WS-PGRADE/gUSE execution service, which does not have provenance capabilities yet.

A provenance data collector was developed to gather provenance information about the scientific experiments performed using the Neuroscience gateway. For each workflow execution it collects data related to the jobs, their inputs and output results, users in charge of the experiments, and dependency relationships among these data. The collector follows a similar approach to our previous implementations [51, 52], deploying *PROV-man* to gather provenance information and organize it according to the experiment context. Figure 11 illustrates two use case scenarios of the provenance collector, namely gUSE/WS-PGRADE and Neuroscience gateway where detailed information about executed workflows are gathered from gUse and NSG databases, as well as from the log files generated by the jobs executed on the DCIs.

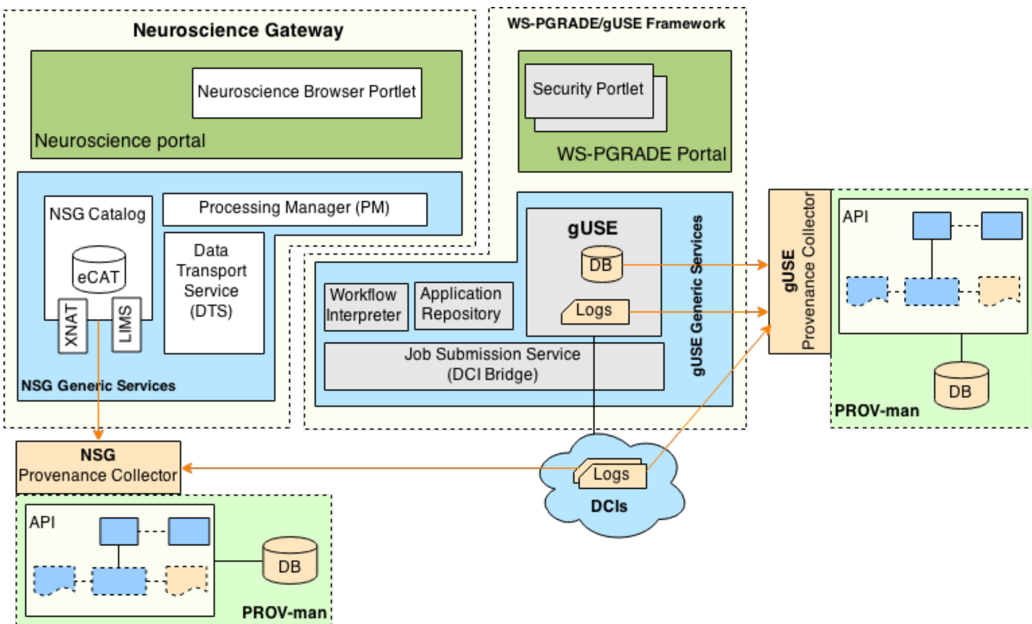

Figure 11: Architecture of the provenance data collector for the Neuroscience gateway. Only components related to provenance are depicted.

The mapping of workflows execution data to PROV concepts is straightforward for both use cases. Each workflow/experiment maps to a *Document* in the *PROV-man* database, jobs are mapped to *Activities*, input/output data to *Entities* and users are mapped to *Agents*. The most important *Relations* linking the input data to the output results in each experiment are *used* and *wasGeneratedBy*. Descriptive details documenting the properties of the core data types and relationships are mapped into the *PROV-man* database as *Attributes,* such as format, location, and size of input/output data; hostname of computing nodes where the jobs are executed; operating system on the computing nodes; the version of the software tools; etc.

Two main challenges were faced during the data collection and organization using *PROV-man*. The first relates to accessing the log files on the DCIs (Dutch Grid in our case), where the logs are only kept for a short period of time after the job execution. We therefore configured the provenance collector to be triggered as soon a workflow terminates execution. For this reason, for most workflows executed in the past it was not possible to collect details such as start and end time of jobs and computing nodes on which they run. Job start and end time are mapped as direct members of an Activity; however, the final status of a job had to be mapped as an *Attribute*.

The second challenge was to reconstruct the full dependencies between data and jobs in a workflow from the various scattered information sources of gUse and grid job logs. In particular, various operations are needed to correctly link all jobs to their proper input and output data in the context of the workflow. The full dependencies were made possible by identifying the jobs that consume the output generated by other jobs.

To completely avoid both challenges it would be more appropriate to instrument WS-PGRADE/gUse directly to collect such data, following the approach presented in section 4.1.

Enhancing the Neuroscience Gateway with provenance capabilities enabled the automatic collection of provenance information, whenever the scientists used the gateway to analyze and process their data. Currently, the provenance data is used by administrators to generate experiments reports, to draw their execution graphs, and to provide statistics about the executed experiments, used data analysis tools, users in charges, experiments failure/success ratio, execution time, etc. Further exploration of experiment provenance with interactive tools for end users is under development.

5 Discussion

The design and implementation of *PROV-man* can be discussed from different perspectives: technology choices, data model optimization, performance, experiences in adopting the PROV recommendations, and how the *PROV-man* approach fulfills the design requirements.

Technology choices: The choice of a relational DBMS as a back-end for the provenance framework, in combination with Java and Hibernate, guarantees flexibility and openness of the system for selecting the back-end storage. Currently Hibernate supports almost all the RDBMSs, including ORACLE, DB2, MS SQL, MySQL, PostgreSQL, Sybase, Informix, and HSQL. The selection of Java programming language limits the deployment of *PROV-man* into the core of existing software tools (e.g. workflow systems) that are implemented in another language. In such cases, external data collectors can be implemented using *PROV-man*, such as presented in section 4.2. To re-implement *PROV-man* using another programming language, the developer has to select a proper ORM technology, which requires re-designing part the proposed *PROV-man* data model to comply with the chosen technology while keeping the optimizations proposed here. Another solution would be to provide *PROV-man* as a service.

Data model optimization: By using Hibernate ORM constructs, all the PROV relationships could be properly modeled as one *Relation*. We also tested other ORM technologies (namely, Castor JDO [61] and datanucleus [62]), but it was not possible to reach such an optimized data model with them. In our case, each *Relation* contains two foreign keys pointing to the primary keys in the associated core data types; therefore, strict ER modeling would require different tables for each of the PROV *Relations*. Using Hibernate, we were able to use a foreign key in the *Relation* table (*Cause* and *Effect*) to reference to a primary key in more than one table, based on the type of the relationship (*Entity*, *Activity*, *Agent*).

Performance: The deployment of PROV-man within the Neuroscience Gateway, presented in section 4.2, didn't present any performance issues while collecting provenance data related to more than 5000 experiments executed under WS-PGRADE/gUSE framework. The data collection was performed after all experiments are finished or terminated, in such a scenario, the process takes few miliseconds to a second per experiment. However, we didn't test the data collection in cases, where the data is progressively collected during experiments execution, in such a scenario we assume that some performance issues may occur in distributed environments involving large number of experiments executed simultaneously.

Experiences in adopting PROV: With regard to the implementation of the PROV specifications, for our application we noticed that minor modifications could enhance the readability of the standardized provenance data. *Types* and *Roles* of *Agents* and *Relationships* are currently specified as key-value pairs using *Attributes*; however, they are important elements for provenance of scientific experiments and could be better modeled as direct members of these entities. This would make the PROV data model more comprehensive. Similarly, a field *Status* could be added as a member to the *Activity* data type, to indicate its final status (e.g. Done, Failed, Planned).

Design Requirements: With regard to the approach followed by *PROV-man*, we have shown in section 4.2 the flexibility of the *PROV-man* framework and its easy deployment within an existing application. However, it required detailed knowledge about the WS-PGRADE/gUSE framework to identify the pieces of provenance data to be collected and linked according to their proper context. The NSG case also illustrates the compliance of *PROV-man* with the design requirements, defined in section 3.2, in terms of permanent storage of provenance data and support for data sharing using utility functions.

6 Conclusion

In this paper we described the design and implementation of the *PROV-man* framework for management of provenance data. *PROV-man* implements the provenance standard in compliance with the PROV-Constraints and according to the PROV specifications [7]. It has been released as a library that can be directly used from Java applications. To our knowledge, this work is the first to describe a framework to facilitate the capture and storage of PROV-compliant provenance data from generic scientific applications

PROV-man provides methods to create and manipulate provenance data in a consistent manner and ensures the permanent storage of provenance data into a relational database that can be configured and tuned for each application. A set of basic interfaces are provided to serialize and export the provenance data to various data formats. These interfaces can be enhanced with new methods, whenever needed, to better serve the interoperation with emerging applications and eventually, to provide data representation for the PROV family of documents (e.g. PROV-DC, and PROV-LINKS). The open architecture of *PROV-man*, consisting of an API and a configurable database, allows for its straightforward deployment within other software tools to enable or enhance their provenance capabilities. By deploying *PROV-man*, applications can more easily benefit from the advantages of the PROV standard for provenance interoperability.

For example, collaboration project is planned with the developers of WS-PGRADE/gUSE [59] and WSVLAM [63] workflow management systems to implement provenance into their core software using *PROV-man*. The granularity of the provenance data to be collected has to be specified, and, a mapping needs to be defined between workflow and *PROV* concepts. The deployment of *PROV-man* within the workflow management systems will enable the automatic collection of provenance information in interoperable format, whenever scientists use the platform to analyze and process their data.

Acknowledgement

This work is partially supported by the COMMIT program funded by the Netherlands Organization for Scientific Research (NWO) and by the SCI-BUS project, which was funded by European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 28348. The Dutch e-Science Grid is provided by SURFsara and NWO.

7 References

1. J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, and B. Didier, "Multi-Scale Science, Supporting Emerging Practice with Semantically Derived Provenance," in ISWC workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, 2003.
2. J. Widom, "Trio: A System for Integrated Management of Data, Accuracy, and Lineage," in CIDR, 2005.
3. J. Zhao, C. A. Goble, R. Stevens, and S. Bechhofer, "Semantically Linking and Browsing Provenance Logs for Escience," in ICSNW, 2004.
4. Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. SIGMOD Rec., 34(3):31–36, 2005.
5. R. Bose and J. Frew, "Lineage retrieval for scientific data processing: a survey," in ACM Comput. Surv., vol. 37, 2005.
6. L. Moreau, et al. The Open Provenance Model Core Specification (v1.1). Future Generation Computer Systems, vol. 27(6) pp.743-756, June 2011.
7. PROV-Overview: http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/
8. PROV Implementation Report: http://www.w3.org/TR/prov-implementations
9. Huynh, Trung Dong and Moreau, Luc (2014) ProvStore: a public provenance repository. In Proceedings of *5th International Provenance and Annotation Workshop (IPAW'14), Cologne, Germany, 09 - 13 Jun 2014*.
10. Flavio Costa, Vítor Silva, Daniel de Oliveira, Kary A. C. S. Ocaña, Eduardo S. Ogasawara, Jonas Dias, Marta Mattoso: Capturing and querying workflow runtime provenance with PROV: a practical approach. EDBT/ICDT Workshops 2013: 282-289
11. Provenance Wikipedia: http://en.wikipedia.org/wiki/Provenance
12. Peter Buneman, Sanjeev Khanna, and Wang chiew Tan. Why and where: A characterization of data provenance. In In ICDT, pages 316–330. Springer, 2001.
13. DICOM - Digital Imaging and Communications in Medicine: http://dicom.nema.org
14. LIMS: http://en.wikipedia.org/wiki/Laboratory_information_management_system
15. Michael H. Elliott, "Electronic Laboratory Notebooks Enter Mainstream Informatics," Scientific Computing, November 2008
16. J. Lyle and A. Martin. Trusted computing and provenance: better together. In Proceedings of TAPP 2010, Berkeley, CA, USA, 2010. USENIX Association.
17. J. Frew and R. Bose, "Earth System Science Workbench: A Data Management Infrastructure for Earth Science Products," in SSDBM, 2001.
18. Tinga Provenance Service: http://www.tingatech.com
19. M. Greenwood, C. Goble, R. Stevens, J. Zhao, M. Addis, D. Marvin, L. Moreau, and T. Oinn, "Provenance of e-Science Experiments - experience from Bioinformatics," in Proceedings of the UK OST e-Science 2nd AHM, 2003.
20. Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In SIGMOD '06: Proceedings of the 2006 ACM SIGMOD International conference on Management of data, pages 539–550, New York, NY, USA, 2006. ACM.
21. I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," in SSDBM, 2002.
22. Davidson, S.B., Freire, J.: Provenance and scientific workflows: challenges and opportunities. In: SIGMOD Conference, pp. 1345–1350 (2008)
23. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. IEEE Computer 40(12), 26–34 (2007)

24. Jihie Kim, Ewa Deelman, Yolanda Gil, Gaurang Mehta, and Varun Ratnakar. Provenance trails in the wings-pegasus system. Concurr. Comput. : Pract. Exper., 20(5):587–597, 2008.
25. Bertram Ludascher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system: Research articles. Concurr. Comput. : Pract. Exper., 18(10):1039–1065, 2006.
26. P. Missier, K. Belhajjame, J. Zhao, and C. Goble, Data lineage model for Taverna workflows with lightweight annotation requirements, In Proc. of the International Provenance and Annotation Workshop (IPAW), 2008.
27. Moreau, L., Foster, I. (eds.): IPAW 2006. LNCS, vol. 4145. Springer, Heidelberg (2006)
28. The Provenance Challenge Wiki (June 2006), http://twiki.ipaw.info/bin/view/Challenge
29. Miles, S.: Technical summary of the second provenance challenge workshop, King's College (July 2007), http://twiki.ipaw.info/bin/view/Challenge/SecondWorkshopMinutes
30. The Open Provenance Model Luc Moreau, University of Southampton, Juliana Freire, University of Utah, Joe Futrelle, NCSA, Robert E. McGrath, NCSA Jim Myers, NCSA, Patrick Paulson, PNNL December 18, 2007
31. Luc Moreau, Juliana Freire, Joe Futrelle, Robert E. McGrath, Jim Myers, and Patrick Paulson. The Open Provenance Model: An Overview. J. Freire, D. Koop, and L. Moreau (Eds.): IPAW 2008, LNCS 5272, pp. 323–326, 2008. © Springer-Verlag Berlin Heidelberg 2008
32. Shawn Bowers, Timothy McPhillips,Sean Riddle,Manish Kumar Anand,Bertram Ludäscher. Kepler/pPOD: Scientific Workflow and Provenance Support for Assembling the Tree of Life. Lecture Notes in Computer Science Volume 5272, 2008, pp 70-77
33. Paolo Missier, Satya Sahoo, Jun Zhao, Carole Goble, Amit Sheth. Janus: from workflows to semantic provenance and linked open data: *Lecture Notes in Computer Science*, Vol. 6378/2010 (2010), pp. 129-141 Key: citeulike:10019128
34. Y. Simmhan, B. Plale, and D. Gannon, Karma2: Provenance Management for Data Driven Workflows, International Journal of Web Services Research, 5(2):1-22, 2008.
35. C. Silva, J. Freire, and S. Callahan, Provenance for Visualizations: Reproducibility and Beyond, IEEE Computing in Science and Engineering, 9(5):82-29, 2007.
36. Y. Zhao, M. Hategan, B. Cliord, I. Foster, G. vonLaszewski, I. Raicu, T. Stef-Praun, and M. Wilde, Swift: Fast, Reliable, Loosely Coupled Parallel Computation, In Proc. of the International Workshop on Scientific Workflows (SWF), pages 199-206, 2007.
37. PLIER - Provenance Layer Infrastructure for e-Science Resources: http://twiki.ipaw.info/bin/view/OPM/Plier
38. I. Wassink, Matthijs Ooms, P. Neerincx, G. van der Veer, Han Rauwerda, Jack A. M. Leunissen, T. M. Breit, A. Nijholt, P. van der Vet. (2010) *e-BioFlow: improving practical use of workflow systems in bioinformatics.* In: Information Technology in Bio- and Medical Informatics, ITBAM 2010, Sept 1-2, 2010, Bilbao, Spain.
39. Karma provenance collection toolkit: http://d2i.indiana.edu/provenance_karma
40. Chunhyeok Lim , Shiyong Lu , Artem Chebotko , Farshad Fotouhi, Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases, Future Generation Computer Systems, v.27 n.6, p.781-789, June, 2011.
41. Yogesh Simmhan,Roger Barga .Analysis of approaches for supporting the Open Provenance Model: A case study of the Trident workflow workbench Published in:· Journal Future Generation Computer Systems archive Volume 27 Issue 6, June, 2011. Pages 790-796
42. Ashish Gehani and Dawood Tariq, SPADE: Support for Provenance Auditing in Distributed Environments, 13th ACM/IFIP/USENIX International Conference on Middleware, 2012.
43. Rinke Hoekstra and Paul Groth. Linkitup: Link discovery for research data. In Discovery Informatics: AI Takes a Science-Centered View on Big Data, AAAI Fall Symposium Series, 2013.

44. Hoekstra, R. and Groth, P. PROV-O-Viz - Understanding the Role of Activities in Provenance. In Proceedings of *5th International Provenance and Annotation Workshop (IPAW'14) , Cologne, Germany, 09 - 13 Jun 2014.*
45. *Amir Sezavar Keshavarz, Trung Dong Huynh and Luc Moreau.* Provenance for Online Decision Making. In Proceedings of *5th International Provenance and Annotation Workshop (IPAW'14) , Cologne, Germany, 09 - 13 Jun 2014.*
46. *Adianto Wibisono, Peter Bloem, Gerben De Vries, Paul Groth, Adam Belloum and M.* Generating Scientific Documentation for Computational Experiments Using Provenance. In Proceedings of *5th International Provenance and Annotation Workshop (IPAW'14) , Cologne, Germany, 09 - 13 Jun 2014.*
47. *Luiz Gadelha and Marta Mattoso.* Applying Provenance to Protect Attribution in Distributed Computational Scientific Experiments. In Proceedings of *5th International Provenance and Annotation Workshop (IPAW'14) , Cologne, Germany, 09 - 13 Jun 2014.*
48. Wellington Oliveira, Daniel de Oliveira, Vanessa Braganholo. Experiencing PROV-Wf for Provenance Interoperability in SWfMSs. In Proceedings of *5th International Provenance and Annotation Workshop (IPAW'14) , Cologne, Germany, 09 - 13 Jun 2014.*
49. Michael Gerhards, Sascha Skorupa, Volker Sander, Adam Belloum, Dmitry Vasunin, A. Benabdelkader. HisT/PLIER: A two-fold Provenance Approach for Grid-enabled Scientific Workflows using WS-VLAM. In *the 12th IEEE/ACM International Conference on Grid Computing, 22-23 September 2011*, Lyon, France, 2011. ICGC 2011.
50. Michael Gerhards, Sascha Skorupa, Volker Sander, Adam Belloum, Dmitry Vasunin, A. Benabdelkader. Provenance Opportunities for WS-VLAM: An Exploration of an e-Science and an e-Business Approach. Submitted to *the 6th Workshop on Workflows in Support of Large-Scale Science, November 12-18, 2011*, Seattle, 2011. - WSLSS 2011
51. A. Benabdelkader,M. Santcroos, S. Madougou, A. H. van Kampen, S. Olabarriaga. A Provenance approach to trace scientific experiments on a grid infrastructure. In *the 7th IEEE International Conference on e-Science, 05-08 December 2011*, Stockholm, Sweden, 2011: 134-141. - e-science 2011
52. Souley Madougou, Shayan Shahand, Mark Santcroos, Barbera D. C. van Schaik, Ammar Benabdelkader, Antoine H. C. van Kampen, Sílvia Delgado Olabarriaga: Characterizing workflow-based activity on a production e-infrastructure using provenance data. Future Generation Comp. Syst. 29(8): 1931-1942 (2013) - FGCS 2013
53. PROV-man software release: http://www.sharp-sys.nl/PROV-man.html
54. G. King, C. Bauer, *"Java Persistence with Hibernate* (Second ed.), "Manning Publications, pp. 880, ISBN 1932394885, November 2006.
55. Graph Visualization Software – Graphviz: www.graphviz.org
56. PROV Model Primer: http://www.w3.org/TR/prov-primer/
57. Shahand S, Benabdelkader A, Jaghoori MM, al Mourabit M, Huguet J, Caan MWA, van Kampen AHC, Olabarriaga SD. A data-centric neuroscience gateway: design, implementation, and experiences. Journal of Concurrency and Computation: Practice and Experience, 27 (2):pp. 489-506, 2015
58. Benabdelkader et al, Collection of provenance data from grid workflow execution using WS-PGRADE/gUse. (initiative https://groups.google.com/forum/#!forum/prov4guse)
59. Kacsuk et al., "WS-PGRADE/gUSE Generic DCI Gateway Frame-work for a Large Variety of User Communities," Journal of Grid Computing , vol. 10, no. 4, pp. 601–630, 2012
60. The SURFsara website, https://www.surfsara.nl
61. Castor 1.3.1 - release and documentation. http://castor.codehaus.org
62. datanucleus open project: http://www.datanucleus.org

63. V. Korkhov, D. Vasyunin, A. Wibisono V. Guevara-Masis, A. Belloum “WS-VLAM: Towards a Scalable Workflow System on the Grid” Workshop on workflows in Support of Large-Scale Science (WORKS 07); In conjunction with HPDC 2007; Monterey Bay, June 2007.
64. COMMIT Project: http://www.commit-nl.nl
65. SCI-BUS - SCIentific gateway Based User Support: http://www.sci-bus.eu

5 Suplementary Material:

5.1 PROV Data Model: Complete ER Schema

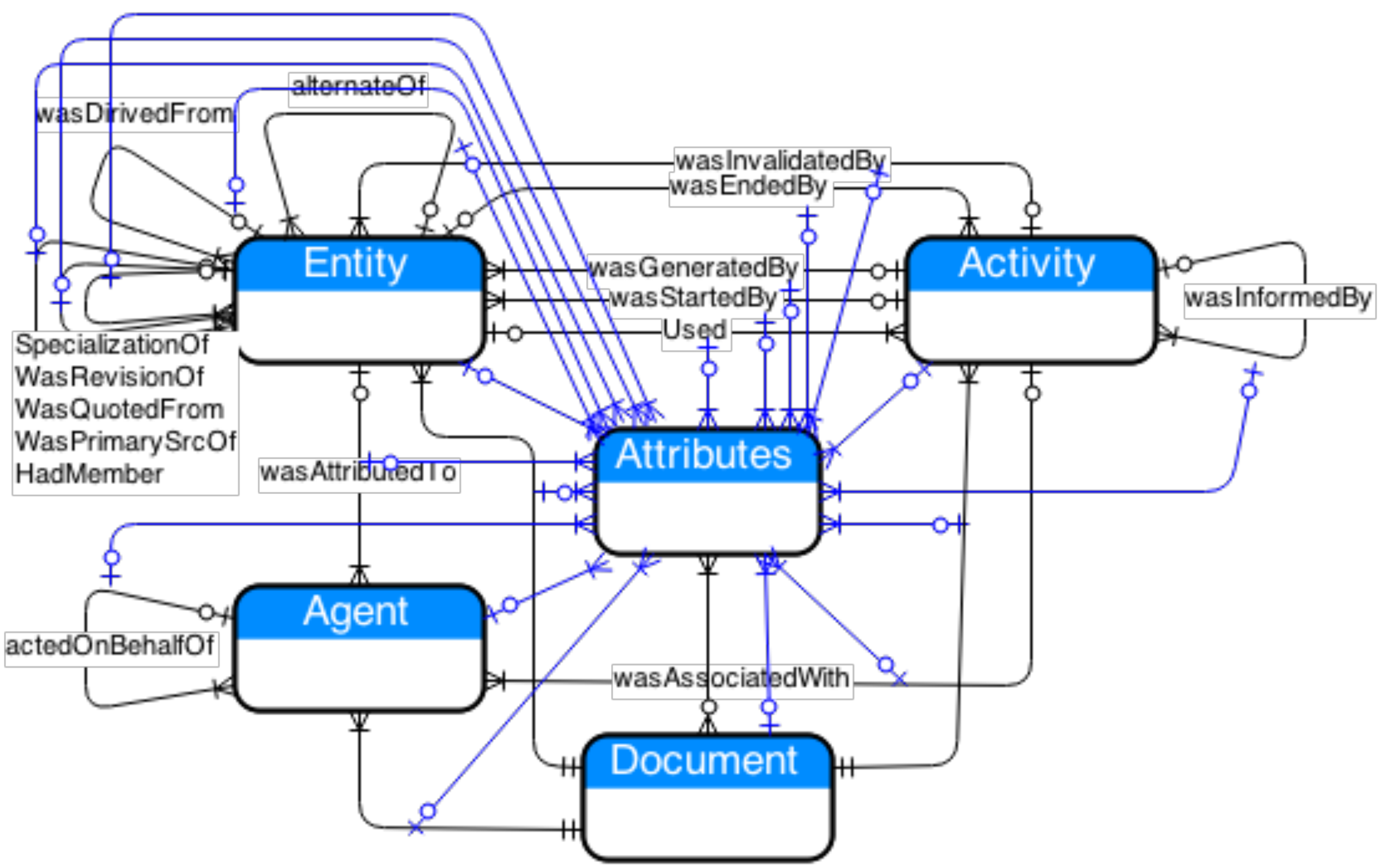

Figure 12: PROV-DM core data types with their complete set of relationships.