W3C Recommendation

PROV-DM: The PROV Data Model

W3C Recommendation 30 April 2013

This version:
http://www.w3.org/TR/2013/REC-prov-dm-20130430/
Latest published version:
http://www.w3.org/TR/prov-dm/
Implementation report:
http://www.w3.org/TR/2013/NOTE-prov-implementations-20130430/
Previous version:
http://www.w3.org/TR/2013/PR-prov-dm-20130312/ (color-coded diff)

Editors:
Luc Moreau, University of Southampton
Paolo Missier, Newcastle University
Contributors:
Khalid Belhajjame, University of Manchester
Reza B'Far, Oracle Corporation
James Cheney, University of Edinburgh
Sam Coppens, iMinds - Ghent University
Stephen Cresswell, legislation.gov.uk
Yolanda Gil, Invited Expert
Paul Groth, VU University of Amsterdam
Graham Klyne, University of Oxford
Timothy Lebo, Rensselaer Polytechnic Institute
Jim McCusker, Rensselaer Polytechnic Institute
Simon Miles, Invited Expert
James Myers, Rensselaer Polytechnic Institute
Satya Sahoo, Case Western Reserve University
Curt Tilmes, National Aeronautics and Space Administration

Please refer to the **errata** for this document, which may include some normative corrections.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

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. PROV-DM is the conceptual data model that forms a basis for the W3C provenance (PROV) family of specifications. PROV-DM distinguishes core structures, forming the essence of provenance information, from extended structures catering for more specific uses of provenance. PROV-DM is organized in six components, respectively dealing with: (1) entities and activities, and the time at which they were created, used, or ended; (2) derivations of entities from entities; (3) agents bearing responsibility for entities that were generated and activities that happened; (4) a notion of bundle, a mechanism to support provenance of provenance; (5) properties to link entities that refer to the same thing; and, (6) collections forming a logical structure for its members.

This document introduces the provenance concepts found in PROV and defines PROV-DM types and relations. The PROV data model is domain-agnostic, but is equipped with extensibility points allowing domain-specific information to be included.

Two further documents complete the specification of PROV-DM. First, a companion document specifies the set of constraints that provenance should follow. Second, a separate document describes a provenance notation for expressing instances of provenance for human consumption; this notation is used in examples in this document.

The PROV Document Overview describes the overall state of PROV, and should be read before other PROV documents.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

PROV Family of Documents

This document is part of the PROV family of documents, a set of documents defining various aspects that are necessary to achieve the vision of inter-operable interchange of provenance information in heterogeneous environments such as the Web. These documents are listed below. Please consult the [PROV-OVERVIEW] for a guide to reading these documents.

- PROV-OVERVIEW (Note), an overview of the PROV family of documents [PROV-OVERVIEW];
- PROV-PRIMER (Note), a primer for the PROV data model [PROV-PRIMER];
- PROV-O (Recommendation), the PROV ontology, an OWL2 ontology allowing the mapping of the PROV data model to RDF [PROV-O];
- PROV-DM (Recommendation), the PROV data model for provenance (this document);
- PROV-N (Recommendation), a notation for provenance aimed at human consumption [PROV-N];
- PROV-CONSTRAINTS (Recommendation), a set of constraints applying to the PROV data model [PROV-CONSTRAINTS];
- PROV-XML (Note), an XML schema for the PROV data model [PROV-XML];
- PROV-AQ (Note), mechanisms for accessing and querying provenance [PROV-AQ];
- PROV-DICTIONARY (Note) introduces a specific type of collection, consisting of key-entity pairs [PROV-DICTIONARY];
- PROV-DC (Note) provides a mapping between PROV-O and Dublin Core Terms [PROV-DC];
- PROV-SEM (Note), a declarative specification in terms of first-order logic of the PROV data model [PROV-SEM];
- PROV-LINKS (Note) introduces a mechanism to link across bundles [PROV-LINKS].

Endorsed By W3C

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

Please Send Comments

This document was published by the Provenance Working Group as a Recommendation. If you wish to make comments regarding this document, please send them to public-prov-comments@w3.org (subscribe, archives). All comments are welcome.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

1. Introduction
    1.1 Compliance with this Document
    1.2 Structure of this Document
    1.3 Notational Conventions
    1.4 Namespaces
2. PROV Overview
    2.1 PROV Core Structures
        2.1.1 Entity and Activity
        2.1.2 Derivation
        2.1.3 Agents and Responsibility
    2.2 PROV Extended Structures
        2.2.1 Mechanisms to Define Extended Structures
            2.2.1.1 Subtyping
            2.2.1.2 Expanded Relations
            2.2.1.3 Optional Identification
            2.2.1.4 Further Relations
        2.2.2 Provenance of Provenance
        2.2.3 Collections
    2.3 Modular Organization
3. The Provenance Notation
4. Illustration of PROV-DM by an Example
    4.1 Example: The Authors View

- 4.2 Example: The Process View
- 4.3 Example: Attribution of Provenance

5. PROV-DM Types and Relations
- 5.1 Component 1: Entities and Activities
  - 5.1.1 Entity
  - 5.1.2 Activity
  - 5.1.3 Generation
  - 5.1.4 Usage
  - 5.1.5 Communication
  - 5.1.6 Start
  - 5.1.7 End
  - 5.1.8 Invalidation
- 5.2 Component 2: Derivations
  - 5.2.1 Derivation
  - 5.2.2 Revision
  - 5.2.3 Quotation
  - 5.2.4 Primary Source
- 5.3 Component 3: Agents, Responsibility, and Influence
  - 5.3.1 Agent
  - 5.3.2 Attribution
  - 5.3.3 Association
  - 5.3.4 Delegation
  - 5.3.5 Influence
- 5.4 Component 4: Bundles
  - 5.4.1 Bundle constructor
  - 5.4.2 Bundle Type
- 5.5 Component 5: Alternate Entities
  - 5.5.1 Specialization
  - 5.5.2 Alternate
- 5.6 Component 6: Collections
  - 5.6.1 Collection
  - 5.6.2 Membership
- 5.7 Further Elements of PROV-DM
  - 5.7.1 Identifier
  - 5.7.2 Attribute
    - 5.7.2.1 prov:label
    - 5.7.2.2 prov:location
    - 5.7.2.3 prov:role
    - 5.7.2.4 prov:type
    - 5.7.2.5 prov:value
  - 5.7.3 Value
  - 5.7.4 Namespace Declaration
  - 5.7.5 Qualified Name

6. PROV-DM Extensibility Points

7. Creating Valid Provenance

A. Cross-References to PROV-O and PROV-N

B. Change Log
- B.1 Changes since Proposed Recommendation
- B.2 Changes since Candidate Recommendation
- B.3 Changes since Last Call

C. Acknowledgements

D. References
- D.1 Normative references
- D.2 Informative references

1. Introduction

For the purpose of this specification, ***provenance***◊ is defined as a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing. In particular, the provenance of information is crucial in deciding whether information is to be trusted, how it should be integrated with other diverse information sources, and how to give credit to its originators when reusing it. In an open and inclusive environment such as the Web, where users find information that is often contradictory or questionable, provenance can help those users to make trust judgements.

We present the PROV data model, PROV-DM, a generic data model for provenance that allows domain and application specific representations of provenance to be translated into such a data model and *interchanged* between systems. Thus, heterogeneous systems can export their native provenance into such a core data model, and applications that need to make sense of provenance can then import it, process it, and reason over it.

The PROV data model distinguishes *core structures* from *extended structures*: core structures form the essence of provenance information, and are commonly found in various domain-specific vocabularies that deal with provenance or similar kinds of information [Mappings]. Extended structures enhance and refine core structures with more expressive capabilities to cater for more advanced uses of provenance. The PROV data model, comprising both core and extended structures, is a domain-agnostic model, but with clear extensibility points allowing further domain-specific and application-specific extensions to be defined.

The PROV data model has a modular design and is structured according to six components covering various facets of provenance:

- component 1: entities and activities, and the time at which they were created, used, or ended;
- component 2: derivations of entities from others;
- component 3: agents bearing responsibility for entities that were generated and activities that happened;
- component 4: bundles, a mechanism to support provenance of provenance;
- component 5: properties to link entities that refer to the same thing;
- component 6: collections forming a logical structure for its members.

This specification presents the concepts of the PROV data model, and provenance types and relations, without specific concern for how they are applied. With these, it becomes possible to write useful provenance, and publish or embed it alongside the data it relates to.

However, if something about which provenance is expressed is subject to change, then it is challenging to express its provenance precisely (e.g. the data from which a daily weather report is derived changes from day to day). This is addressed in a companion specification [PROV-CONSTRAINTS] by proposing formal constraints on the way that provenance is related to the things it describes (such as the use of attributes, temporal information and specialization of entities), and additional conclusions that are valid to infer.

1.1 Compliance with this Document

For the purpose of compliance, the normative sections of this document are Section 1.1, Section 1.3, Section 5., and Appendix A.

- Information in tables is normative if it appears in a normative section.
- All figures (including UML diagrams) are informative.
- Text in boxes labeled "Example" is informative.

1.2 Structure of this Document

This section is non-normative.

Section 2 provides an overview of the PROV data model, distinguishing a core set of types and relations, commonly found in provenance, from extended structures catering for more specific uses. It also introduces a modular organization of the data model in components.

Section 3 overviews the Provenance Notation used to illustrate examples of provenance.

Section 4 illustrates how the PROV data model can be used to express the provenance of a report published on the Web.

Section 5 provides the definitions of PROV concepts, structured according to six components.

Section 6 summarizes PROV-DM extensibility points.

Section 7 introduces the idea that constraints can be applied to the PROV data model to validate provenance; these are covered in the companion specification [PROV-CONSTRAINTS].

1.3 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Examples throughout this document use the PROV-N Provenance Notation, briefly introduced in Section 3 and specified fully in a separate document [PROV-N].

1.4 Namespaces

This section is non-normative.

The following namespaces prefixes are used throughout this document.

Table 1◊: Prefix and Namespaces used in this specification

prefix	namespace IRI	definition

prov	http://www.w3.org/ns/prov#	The PROV namespace (see Section 5.7.4)
xsd	http://www.w3.org/2000/10/XMLSchema#	XML Schema Namespace [XMLSCHEMA11-2]
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF namespace [RDF-CONCEPTS]
(others)	(various)	All other namespace prefixes are used in examples only. In particular, IRIs starting with "http://example.com" represent some application-dependent IRI [RFC3987]

2. PROV Overview

This section is non-normative.

This section introduces provenance concepts with informal explanations and illustrative examples. PROV distinguishes *core structures*, forming the essence of provenance, from *extended structures* catering for more specific uses of provenance. Core and extended structures are respectively presented in Section 2.1 and Section 2.2. Furthermore, the PROV data model is organized according to components, which form thematic groupings of concepts (see Section 2.3). A *provenance description* is an instance of a provenance structure, whether core or extended, described below.

2.1 PROV Core Structures

This section is non-normative.

At its core, provenance describes the use and production of *entities* by *activities*, which may be influenced in various ways by *agents*. These core types and their relationships are illustrated by the UML diagram of Figure 1.

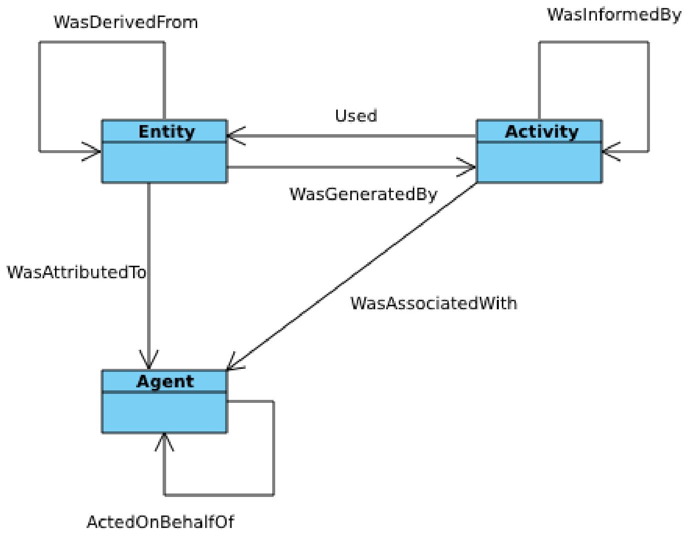

Figure 1◊: PROV Core Structures (Informative)

The concepts found in the core of PROV are introduced in the rest of this section. They are summarized in Table 2, where they are categorized as type or relation. The first column lists concepts, the second column indicates whether a concept maps to a type or a relation, whereas the third column contains the corresponding name, as it appears in Figure 1. Names of relations have a verbal form in the past tense to express what happened in the past, as opposed to what may or will happen. In the core of PROV, all relations are binary.

Table 2◊: Mapping of PROV core concepts to types and relations

PROV Concepts	PROV-DM types or relations	Name	Overview
Entity	PROV-DM Types	Entity	Section 2.1.1
Activity		Activity	Section 2.1.1
Agent		Agent	Section 2.1.3
Generation	PROV-DM Relations	WasGeneratedBy	Section 2.1.1
Usage		Used	Section 2.1.1
Communication		WasInformedBy	Section 2.1.1
Derivation		WasDerivedFrom	Section 2.1.2
Attribution		WasAttributedTo	Section 2.1.3
Association		WasAssociatedWith	Section 2.1.3
Delegation		ActedOnBehalfOf	Section 2.1.3

Hide Concept Examples

2.1.1 Entity and Activity

This section is non-normative.

In PROV, things we want to describe the provenance of are called *entities* and have some fixed aspects. The term "things" encompasses a broad diversity of notions, including digital objects such as a file or web page, physical things such as a mountain, a building, a printed book, or a car as well as abstract concepts and ideas.

*An **entity** is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.* [Detailed specification]

Example 1◊

An entity may be the document at IRI http://www.bbc.co.uk/news/science-environment-17526723, a file in a file system, a car, or an idea.

*An **activity** is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.* [Detailed specification] Just as entities cover a broad range of notions, activities can cover a broad range of notions: information processing activities may for example move, copy, or duplicate digital entities; physical activities can include driving a car between two locations or printing a book.

Example 2◊

An activity may be the publishing of a document on the Web, sending a twitter message, extracting metadata embedded in a file, driving a car from Boston to Cambridge, assembling a data set based on a set of measurements, performing a statistical analysis over a data set, sorting news items according to some criteria, running a SPARQL query over a triple store, or editing a file.

Activities and entities are associated with each other in two different ways: activities utilize entities and activities produce entities. The act of utilizing or producing an entity may have a duration. The term 'generation' refers to the completion of the act of producing; likewise, the term 'usage' refers to the beginning of the act of utilizing entities. Thus, we define the following concepts of generation and usage.

***Generation** is the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation.* [Detailed specification]

***Usage** is the beginning of utilizing an entity by an activity. Before usage, the activity had not begun to utilize this entity and could not have been affected by the entity.* [Detailed specification]

Example 3◊

Examples of generation are the completed creation of a file by a program, the completed creation of a linked data set, and the completed publication of a new version of a document.

Example 4◊

Usage examples include a procedure beginning to consume an argument, a service starting to read a value on a port, a program beginning to read a configuration file, or the point at which an ingredient, such as eggs, is being added in a baking activity. Usage may entirely consume an entity (e.g. eggs are no longer available after being added to the mix); in contrast, the same entity may be used multiple times, possibly by different activities (e.g. a file on a file system can be read indefinitely).

Example 5◊

Let us consider the activity of driving a car from Boston to Cambridge. One might reasonably ask what entities are used and generated by this activity. This is answered by considering that a single artifact may correspond to several entities; in this case, a car in Boston may be a different entity from the same car in Cambridge. Thus, among other things, an entity "car in Boston" would be used, and a new entity "car in Cambridge" would be generated by this activity of driving. The provenance trace of the car might include: designed in Japan, manufactured in Korea, shipped to Boston USA, purchased by customer, driven to Cambridge, serviced by engineer in Cambridge, etc., all of which might be important information when deciding whether or not it represents a sensible second-hand purchase. Or some of it might alternatively be relevant when trying to determine the truth of a web page reporting a traffic violation involving that car. This breadth of provenance allows descriptions of interactions between physical and digital artifacts.

The generation of an entity by an activity and its subsequent usage by another activity is termed communication.

***Communication** is the exchange of some unspecified entity by two activities, one activity using some entity generated by the other.* [Detailed specification]

Example 6◊

The activity of writing a celebrity article was informed by (a communication instance) the activity of intercepting voicemails.

2.1.2 Derivation

Activities utilize entities and produce entities. In some cases, utilizing an entity influences the creation of another in some way. This notion of 'influence' is captured by derivations, defined as follows.

*A **derivation** is a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity.* [Detailed specification]

Example 7◊

Examples of derivation include the transformation of a relational table into a linked data set, the transformation of a canvas into a painting, the transportation of a work of art from London to New York, and a physical transformation such as the melting of ice into water.

The focus of derivation is on connecting a generated entity to a used entity. While the basic idea is simple, the concept of derivation can be quite subtle: implicit is the notion that the generated entity was affected in some way by the used entity. If an artifact was used by an activity that also generated a new artifact, it does not always follow that the second artifact was derived from the first. In the activity of creating a painting, an artist may have mixed some paint that was never actually applied to the canvas: the painting would typically not be considered a derivation from the unused paint. PROV does not attempt to specify the conditions under which derivations exist; rather, derivation is considered to have been determined by unspecified means. Thus, while a chain of usage and generation is necessary for a derivation to hold between entities, it is not sufficient; some form of influence occurring during the activities involved is also needed.

2.1.3 Agents and Responsibility

For many purposes, a key consideration for deciding whether something is reliable and/or trustworthy is knowing who or what *was reponsible* for its production. Data published by a respected independent organization may be considered more trustworthy than that from a lobby organization; a claim by a well-known scientist with an established track record may be more believed than a claim by a new student; a calculation performed by an established software library may be more reliable than by a one-off program.

*An **agent** is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.* [Detailed specification] An agent may be a particular type of entity or activity. This means that the model can be used to express provenance of the agents themselves.

Example 8◊

Software for checking the use of grammar in a document may be defined as an agent of a document preparation activity; one can also describe its provenance, including for instance the vendor and the version history. A site selling books on the Web, the services involved in the processing of orders, and the companies hosting them are also agents.

Agents can be related to entities, activities, and other agents.

***Attribution** is the ascribing of an entity to an agent.* [Detailed specification]

Example 9◊

A blog post can be attributed to an author, a mobile phone to its manufacturer.

Agents are defined as having some kind of responsibility for activities.

*An activity **association** is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity.* [Detailed specification]

Example 10◊

Examples of association between an activity and an agent are:

- creation of a web page under the guidance of a designer;
- various forms of participation in a panel discussion, including audience member, panelist, or panel chair;
- a public event, sponsored by a company, and hosted by a museum;

***Delegation** is the assignment of authority and responsibility to an agent (by itself or by another agent) to carry out a specific activity as a delegate or representative, while the agent it acts on behalf of retains some responsibility for the outcome of the delegated work.* [Detailed specification] The nature of this relation is intended to be broad, including contractual relation, but also altruistic initiative by the representative agent.

Example 11◊

A student publishing a web page describing an academic department could result in both the student and the department being agents associated with the activity. It may not matter which actual student published a web page, but it may matter significantly that the department told the student to put up the web page.

2.2 PROV Extended Structures

While the core of PROV focuses on essential provenance structures commonly found in provenance descriptions, extended structures are designed to support more advanced uses of provenance. The purpose of this section is twofold. First, mechanisms to specify these extended structures are introduced. Second, two further kinds of provenance structures are overviewed: they cater for provenance of provenance and collections, respectively.

2.2.1 Mechanisms to Define Extended Structures

Extended structures are defined by a variety of mechanisms outlined in this section: subtyping, expanded relations, optional identification, and new relations.

2.2.1.1 Subtyping

This section is non-normative.

Subtyping can be applied to core types. For example, a software agent is special kind of agent, defined as follows.

*A **software agent** is running software.*

Subtyping can also be applied to core relations. For example, a revision is a special kind of derivation, defined as follows.

*A **revision** is a derivation for which the resulting entity is a revised version of some original.*

2.2.1.2 Expanded Relations

Section 2.1 shows that seven concepts are mapped to binary relations in the core of PROV. However, some advanced uses of these concepts cannot be captured by a binary relation, but require relations to be expanded to n-ary relations.

Indeed, binary relations are actually shorthands that can be 'opened up' by applications and filled in with further application details. For example, derivation is a very high level relationship between two entities: an application may decide to 'open up' that relationship in an expanded relation that describes how an entity was derived from another by virtue of listing the generation, usage, and activity involved in the derivation relationship. Applications are free to decide which level of granularity they want describe, and PROV gives them the way to do that.

To illustrate expanded relations, we revisit the concept of association, introduced in Section 2.1.3 (full definition of the expanded association can be found in Section 5.3.3). Agents may rely on *plans*, i.e. sets of actions or steps, to achieve their goals in the context of an activity. Hence, an expanded form of association relation allows for a plan to be specified. Plan is defined by subtyping and full association by an expanded relation, as follows.

*A **plan** is an entity that represents a set of actions or steps intended by one or more agents to achieve some goals.*

*An activity **association** is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity. It further allows for a plan to be specified, which is the plan intended by the agent to achieve some goals in the context of this activity.*

There exist no prescriptive requirements on the nature of plans, their representation, the actions or steps they consist of, or their intended goals. Since plans may evolve over time, it may become necessary to track their provenance, so plans themselves are entities. Representing the plan explicitly in the provenance can be useful for various tasks: for example, to validate the execution as represented in the provenance record, to manage expectation failures, or to provide explanations.

Example 12◊

An example of association between an activity and an agent involving a plan is: an XSLT transform (an activity) launched by a user (an agent) based on an XSL style sheet (a plan).

2.2.1.3 Optional Identification

Some concepts exhibit both a core use, expressed as binary relation, and an extended use, expressed as n-ary relation. In some cases, mapping the concept to a relation, whether binary or n-ary, is not sufficient: instead, it may be required to identify an instance of such concept. In those cases, PROV allows for an optional identifier to be expressed to identify an instance of an association between two or more elements. This optional identifier can then be used to refer to an instance as part of other concepts.

Example 13◊

A service may read a same configuration file on two different occasions. Each usage can be identifed by its own identifier, allowing them to be distinguished.

2.2.1.4 Further Relations

Finally, PROV supports further relations that are not subtypes or expanded versions of existing relations (such as specialization, alternate).

2.2.2 Provenance of Provenance

*A **bundle** is a named set of provenance descriptions, and is itself an entity, so allowing provenance of provenance to be expressed.*

For users to decide whether they can place their trust in something, they may want to analyze its provenance, but also determine the agent its provenance is attributed to, and when it was generated. In other words, users need to be able to determine the provenance of provenance. Hence, provenance is also regarded as an entity (of type Bundle), by which provenance of provenance can then be expressed.

Example 14◊

In a decision making situation, decision makers may be presented with the same piece of knowledge, issued by multiple sources. In order to validate this piece of knowledge, decision makers can consider its provenance, but also the provenance of its provenance, which may help determine whether it can be trusted.

2.2.3 Collections

*A **collection** is an entity that provides a structure to some constituents that must themselves be entities. These constituents are said to be **member** of the collections.* Many different types of collections exist, such as *sets*, *dictionaries*, or *lists*. Using Collections, one can express the provenance of the collection itself in addition to that of the members.

Example 15◊

An example of collection is an archive of documents. Each document has its own provenance, but the archive itself also has some provenance: who maintained it, which documents it contained at which point in time, how it was assembled, etc.

2.3 Modular Organization

Besides the separation between core and extended structures, PROV-DM is further organized according to components, grouping concepts in a thematic manner.

Table 3 enumerates the six components, five of which have already been implicitly overviewed in this section. All components contain extended structures, whereas only the first three contain core structures.

Table 3◊: Components Overview

	Component	Core Structures	Overview	Specification	Description
1	Entities and Activities	✓	2.1.1	5.1	about entities and activities, and their interrelations
2	Derivation	✓	2.1.2	5.2	about derivation and its subtypes
3	Agent and Responsibility	✓	2.1.3	5.3	about agents and concepts ascribing responsibility to them
4	Bundles		2.2.2	5.4	about bundles, a mechanism to support provenance of provenance
5	Alternate		—	5.5	about relations linking entities referring to the same thing
6	Collections		2.2.3	5.6	about collections

3. The Provenance Notation

This section is non-normative.

To illustrate the application of PROV concepts to a concrete example (see Section 4) and to provide examples of concepts (see Section 5), we introduce PROV-N, a notation for writing instances of the PROV data model. For full details and for a normative reference, the reader is referred to the companion specification [PROV-N]. PROV-N is a notation aimed at human consumption, with the following characteristics:

- PROV-N expressions adopt a *functional notation* consisting of a name and a list of arguments in parentheses.
- The interpretation of PROV-N arguments is defined according to their *position* in the list of arguments. This convention allows for a compact notation.
- The PROV data model defines *identifiers* as qualified names; in PROV-N, they are expressed as a local name optionally preceded of a prefix and a colon.
- PROV-N *optional arguments* need not be specified: the general rule for optional arguments is that, if none of them are used in the expression, then they are simply omitted, resulting in a simpler expression. However, it may be the case that only some of the optional arguments need to be specified. Because the position of the arguments in the expression matters, in this case, an additional marker must be used to indicate that a particular term is not available. The syntactic marker '-' is used for this purpose.
- Most expressions include an identifier and a set of attribute-value pairs; both are optional unless otherwise specified. By convention, the identifier occurs in the *first position*, and the set of attribute-value pairs in the *last position*. Consistent with the convention on arguments, the marker '-' can be used when the identifier is not available, or can be omitted altogether with no ambiguity arising. To further disambiguate expressions that contain an optional identifier, the optional identifier or marker must be followed by ';'.

Example 16◊

An activity with identifier a1 and an attribute type with value createFile.

```
activity(a1, [ prov:type="createFile" ])
```

Two entities with identifiers e1 and e2.

```
entity(e1)
entity(e2)
```

The activity a1 used e1, and e2 was generated by a1.

```
used(a1, e1)
wasGeneratedBy(e2, a1)
```

The same descriptions, but with an explicit identifier u1 for the usage, and the syntactic marker '-' to mark the absence of identifier in the generation. Both are followed by ';'.

```
used(u1; a1, e1)
wasGeneratedBy(-; e2, a1)
```

4. Illustration of PROV-DM by an Example

This section is non-normative.

Section 2 has introduced some provenance concepts, and how they are expressed as types or relations in the PROV data model. The purpose of this section is to put these concepts into practice in order to express the provenance of some document published on the Web. With this realistic example, PROV concepts are composed together, and a graphical illustration shows a provenance description forming a directed graph, rooted at the entity we want to explain the provenance of, and pointing to the entities, activities, and agents it depended on. This example also shows that, sometimes, multiple provenance descriptions about the same entity can co-exist, which then justifies the need for provenance of provenance.

In this example, we consider one of the many documents published by the World Wide Web Consortium, and describe its provenance. Specifically, we consider the document identified by http://www.w3.org/TR/2011/WD-prov-dm-20111215. Its provenance can be expressed from several perspectives: first, provenance can take the authors' viewpoint; second, it can be concerned with the W3C process. Then, attribution of these two provenance descriptions is provided.

4.1 Example: The Authors View

This section is non-normative.

Description: *A document is edited by some editor, using contributions from various contributors.*

In this perspective, provenance of the document http://www.w3.org/TR/2011/WD-prov-dm-20111215 is concerned with the editing activity as perceived by authors. This kind of information could be used by authors in their CV or in a narrative about this document.

We paraphrase some PROV descriptions, express them with the PROV-N notation, and depict them with a graphical illustration (see Figure 2). Full details of the provenance record can be found here.

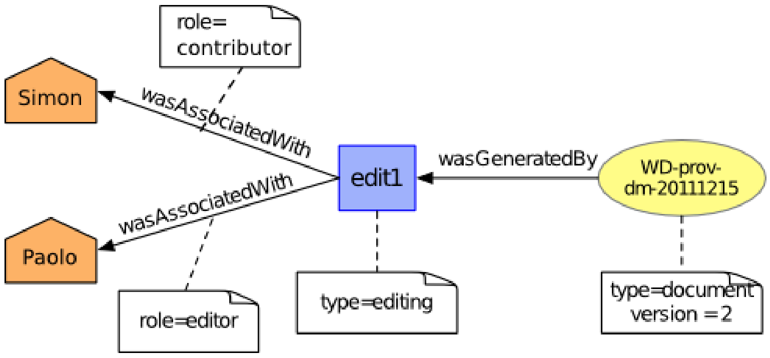

Figure 2: Provenance of a Document (part 1) (Informative)

- There was a document tr:WD-prov-dm-20111215, which from the author's perspective was a document in its second version.

```
entity(tr:WD-prov-dm-20111215, [ prov:type="document", ex:version="2" ])
```

- There was an editing activity.

```
activity(ex:edit1, [ prov:type="editing" ])
```

- The document was generated by the editing activity: this was a Generation. Its time is not specified, hence, the marker '-'.

```
wasGeneratedBy(tr:WD-prov-dm-20111215, ex:edit1, -)
```

- There were some agents.

```
agent(ex:Paolo, [ prov:type='prov:Person' ])
agent(ex:Simon, [ prov:type='prov:Person' ])
```

- Agents were assigned various responsibilities in the editing activity: contributor and editor. The plan the agent relied upon is not specified, hence, the marker '-'.

```
wasAssociatedWith(ex:edit1, ex:Paolo, -, [ prov:role="editor" ])
wasAssociatedWith(ex:edit1, ex:Simon, -, [ prov:role="contributor" ])
```

Provenance descriptions can be *illustrated* graphically. The illustration is not intended to represent all the details of the model, but it is intended to show the essence of a set of provenance descriptions [PROV-LAYOUT]. Therefore, it should not be seen as an alternate notation for expressing provenance.

The graphical illustration takes the form of a graph. Entities, activities and agents are represented as nodes, with oval, rectangular, and pentagonal shapes, respectively. Usage, Generation, Derivation, and Association are represented as directed edges.

Entities are laid out according to the ordering of their generation. We endeavor to show time progressing from left to right. This means that edges for Usage, Generation, Derivation, Association typically point leftwards

4.2 Example: The Process View

Description: *The World Wide Web Consortium publishes documents according to its publication policy. Working drafts are published regularly to reflect the work accomplished by working groups. Every publication of a working draft must be preceded by a "publication request" to the Webmaster. The very first version of a document must also be preceded by a "transition request" to be approved by the W3C director. All working drafts are made available at a unique IRI. In this scenario, we consider two successive versions of a given document, the policy according to which they were published, and the associated requests.*

We describe the kind of provenance record that the WWW Consortium could keep for auditors to check that due processes are followed. All entities involved in this example are Web resources, with well-defined IRIs (some of which refer to archived email messages, available to W3C Members).

- Two versions of a document were involved: `tr:WD-prov-dm-20111215` (second working draft) and `tr:WD-prov-dm-20111018` (first working draft);
- Both `tr:WD-prov-dm-20111215` and `tr:WD-prov-dm-20111018` were published by the WWW Consortium (`w3:Consortium`);
- The publication activity for `tr:WD-prov-dm-20111215` was `ex:act2`;
- The publication activity for `tr:WD-prov-dm-20111018` was `ex:act1`;
- The document `tr:WD-prov-dm-20111215` was derived from `tr:WD-prov-dm-20111018`;
- The publication activity `ex:act1` used a publication request (`email:2011Oct/0141`) and a transition request (`chairs:2011OctDec/0004`);
- The publication activity `ex:act2` used a publication request (`email:2011Dec/0111`);
- Documents were published according to the process rules (`process:rec-advance`), a plan in PROV terminology.

We now paraphrase some PROV descriptions, and express them with the PROV-N notation, and depict them with a graphical illustration (see Figure 3). Full details of the provenance record can be found here.

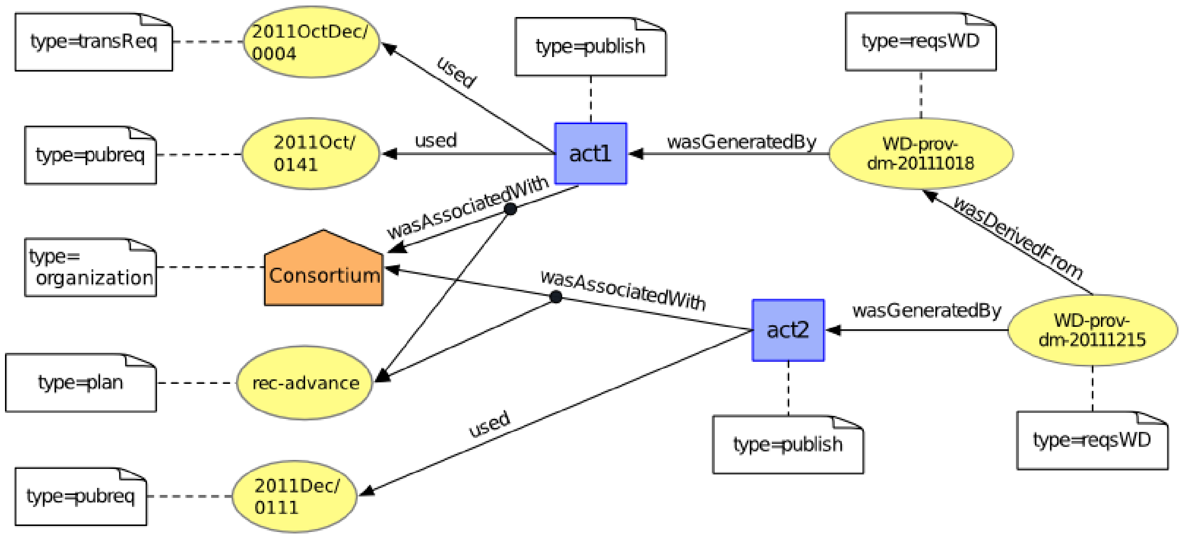

Figure 3◊: Provenance of a Document (part 2) (Informative)

- There was a document, a working draft (rec54:WD), which is an entity so that we can describe its provenance. Similar descriptions exist for all entities.

```
entity(tr:WD-prov-dm-20111215, [ prov:type='rec54:WD' ])
```

- There was a publication activity.

```
activity(ex:act2, [ prov:type="publish" ])
```

- The document was generated by the publication activity: this was a Generation. Its time is not specified, hence, the marker '-'.

```
wasGeneratedBy(tr:WD-prov-dm-20111215, ex:act2, -)
```

- The second draft of the document was derived from the first draft: this was a Derivation.

```
wasDerivedFrom(tr:WD-prov-dm-20111215, tr:WD-prov-dm-20111018)
```

- The activity required a publication request: this was a Usage. Its time is not specified, hence, the marker '-'.

```
used(ex:act2, email:2011Dec/0111, -)
```

- The activity was associated with the Consortium agent, and proceeded according to its publication policy: this is an Association.

  ```
  wasAssociatedWith(ex:act2, w3:Consortium, process:rec-advance)
  ```

  This relation is illustrated in Figure 3 with a multi-edge labelled `wasAssociatedWith` pointing to an agent and entity (representing a plan).

This simple example has shown a variety of PROV concepts, such as Entity, Agent, Activity, Usage, Generation, Derivation, and Association. In this example, it happens that all entities were already Web resources, with readily available IRIs, which we used. We note that some of the resources are public, whereas others have restricted access: provenance statements only make use of their identifiers. If identifiers do not pre-exist, e.g. for activities, then they can be generated, for instance `ex:act2`, occurring in the namespace identified by prefix `ex`. We note that the IRI scheme developed by W3C is particularly suited for expressing provenance of these documents, since each IRI denotes a specific version of a document. It then becomes easy to relate the various versions with PROV relations. We note that an Association is a ternary relation (represented by a multi-edge labeled wasAssociatedWith) from an activity to an agent and a plan.

4.3 Example: Attribution of Provenance

The two previous sections offer two different perspectives on the provenance of a document. PROV allows for multiple sources to provide the provenance of a subject. For users to decide whether they can place their trust in the document, they may want to analyze its provenance, but also determine who the provenance is attributed to, and when it was generated, etc. In other words, we need to be able to express the provenance of provenance.

PROV-DM offers a construct to name a bundle of provenance descriptions (full details: ex:author-view).

```
bundle ex:author-view

  agent(ex:Paolo,   [ prov:type='prov:Person' ])
  agent(ex:Simon,   [ prov:type='prov:Person' ])


...

endBundle
```

Likewise, the process view can be expressed as a separate named bundle (full details: ex:process-view).

```
bundle ex:process-view

   agent(w3:Consortium, [ prov:type='prov:Organization' ])

...

endBundle
```

To express their respective provenance, these bundles must be seen as entities, and all PROV constructs are now available to express their provenance. In the example below, `ex:author-view` is attributed to the agent `ex:Simon`, whereas `ex:process-view` to `w3:Consortium`.

```
entity(ex:author-view, [ prov:type='prov:Bundle' ])
wasAttributedTo(ex:author-view, ex:Simon)

entity(ex:process-view, [ prov:type='prov:Bundle' ])
wasAttributedTo(ex:process-view, w3:Consortium)
```

5. PROV-DM Types and Relations

Provenance concepts, expressed as PROV-DM types and relations, are organized according to six components that are defined in this section. The components and their dependencies are illustrated in Figure 4. A component that relies on concepts defined in another is displayed above it in the figure. So, for example, component 5 (alternate) depends on concepts defined in component 4 (bundles), itself dependent on concepts defined in component 1 (entity and activity).

- **Component 1: entities and activities.** The first component consists of entities, activities, and concepts linking them, such as generation, usage, start, end. The first component is the only one comprising time-related concepts.
- **Component 2: derivations.** The second component is formed with derivations and derivation subtypes.
- **Component 3: agents, responsibility, and influence.** The third component consists of agents and concepts ascribing responsibility to agents.
- **Component 4: bundles.** The fourth component is concerned with bundles, a mechanism to support provenance of provenance.
- **Component 5: alternate.** The fifth component consists of relations linking entities referring to the same thing.
- **Component 6: collections.** The sixth component is about collections.

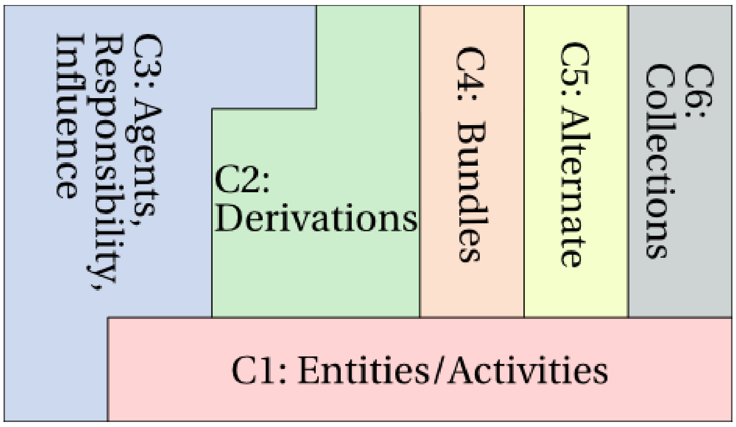

Figure 4◊: PROV-DM Components (Informative)

While not all PROV-DM relations are binary, they all involve two primary elements. Hence, Table 4 indexes all relations (except wasInfluencedBy) according to their two primary elements (referred to as subject and object). The table adopts the same color scheme as Figure 4, allowing components to be readily identified. Relation names appearing in bold correspond to the core structures introduced in Section 2.1.

Table 4◊: PROV-DM Relations At a Glance

		Object					
		Entity		Activity		Agent	
Subject	Entity	**WasDerivedFrom** Revision Quotation PrimarySource AlternateOf SpecializationOf HadMember		**WasGeneratedBy** WasInvalidatedBy	*R* *T* *L*	**WasAttributedTo**	
	Activity	**Used** WasStartedBy WasEndedBy	*R* *T* *L*	**WasInformedBy**		**WasAssociatedWith**	*R*
	Agent	—		—		**ActedOnBehalfOf**	

The letters 'R' and 'L' appearing in the right-hand side of some cells of Table 4 indicate that attributes prov:role (Section 5.7.2.3) and prov:location (Section 5.7.2.2) are permitted for these relations. The letter 'T' indicates an OPTIONAL time is also permitted.

Some PROV-DM relations are not binary and involve extra optional element. They are summarized in Table 5 grouping secondary objects, according to their type. The table also adopts the same color scheme as Figure 4, allowing components to be readily identified. None of these relations correspond to the core structures introduced in Section 2.1.

Table 5◊: Secondary optional elements in PROV-DM Relations

		Secondary Object		
		Entity	Activity	Agent
Subject	Entity	—	WasDerivedFrom (activity)	—
	Activity	WasAssociatedWith (plan)	WasStartedBy (starter) WasEndedBy (ender)	—
	Agent	—	ActedOnBehalfOf (activity)	—

Table 6 is a complete index of all the types and relations of PROV-DM, color-coded according to the component they belong to. In the first column, concept names link to their informal definition, whereas, in the second column, representations link to the information used to represent the concept. Concept names appearing in bold in the first column are the core structures introduced in Section 2.1. Likewise, these core structures have their names and parameters highlighted in bold in the second column (prov-n representation); expanded structures are not represented with a bold font.

Table 6◊: PROV-DM Types and Relations

Type or Relation Name	**Representation in the PROV-N notation**	**Component**
Entity	**entity(id, [attr1=val1, ...])**	
Activity	**activity(id, st, et, [attr1=val1, ...])**	
Generation	**wasGeneratedBy**(id;**e**,**a**,**t**,attrs)	

Usage	**used(**id;**a,e**,t,attrs**)**	Component 1: Entities/Activities
Communication	**wasInformedBy(**id;**a2,a1**,attrs**)**	
Start	wasStartedBy(id;a2,e,a1,t,attrs)	
End	wasEndedBy(id;a2,e,a1,t,attrs)	
Invalidation	wasInvalidatedBy(id;e,a,t,attrs)	
Derivation	**wasDerivedFrom(**id; **e2, e1**, a, g2, u1, attrs**)**	Component 2: Derivations
Revision	... prov:type='prov:Revision' ...	
Quotation	... prov:type='prov:Quotation' ...	
Primary Source	... prov:type='prov:PrimarySource' ...	
Agent	**agent(id, [attr1=val1, ...])**	Component 3: Agents, Responsibility, Influence
Attribution	**wasAttributedTo(**id;**e,ag**,attr**)**	
Association	**wasAssociatedWith(**id;**a,ag**,pl,attrs**)**	
Delegation	**actedOnBehalfOf(**id;**ag2,ag1**,a,attrs**)**	
Plan	... prov:type='prov:Plan' ...	
Person	... prov:type='prov:Person' ...	
Organization	... prov:type='prov:Organization' ...	
SoftwareAgent	... prov:type='prov:SoftwareAgent' ...	
Influence	wasInfluencedBy(id;e2,e1,attrs)	
Bundle constructor	bundle id description_1 ... description_n endBundle	Component 4: Bundles
Bundle type	... prov:type='prov:Bundle' ...	
Alternate	alternateOf(alt1, alt2)	Component 5: Alternate
Specialization	specializationOf(infra, supra)	
Collection	... prov:type='prov:Collection' ...	Component 6: Collections
EmptyCollection	... prov:type='prov:EmptyCollection' ...	
Membership	hadMember(c,e)	

In the rest of the section, each type and relation is defined informally, followed by a summary of the information used to represent the concept, and illustrated with PROV-N examples.

5.1 Component 1: Entities and Activities

The first component of PROV-DM is concerned with entities and activities, and their interrelations: Used (Usage), WasGeneratedBy (Generation), WasStartedBy (Start), WasEndedBy (End), WasInvalidatedBy (Invalidation), and WasInformedBy (Communication). Figure 5 uses UML to depict the first component. Core structures are displayed in the yellow area, consisting of two classes (Entity, Activity) and three binary associations between them: Used (Usage), WasGeneratedBy (Generation), and WasInformedBy (Communication). The rest of the figure displays extended structures, including UML association classes (see [UML], section 7.3.4, p. 42), represented in gray, to express expanded n-ary relations for Used (Usage), WasGeneratedBy (Generation), WasInvalidatedBy (Invalidation), WasStartedBy (Start), WasEndedBy (End). The figure also makes explicit associations with *time* for these concepts (time being marked with the primitive stereotype). When not specified, cardinality is assumed to be 0..*.

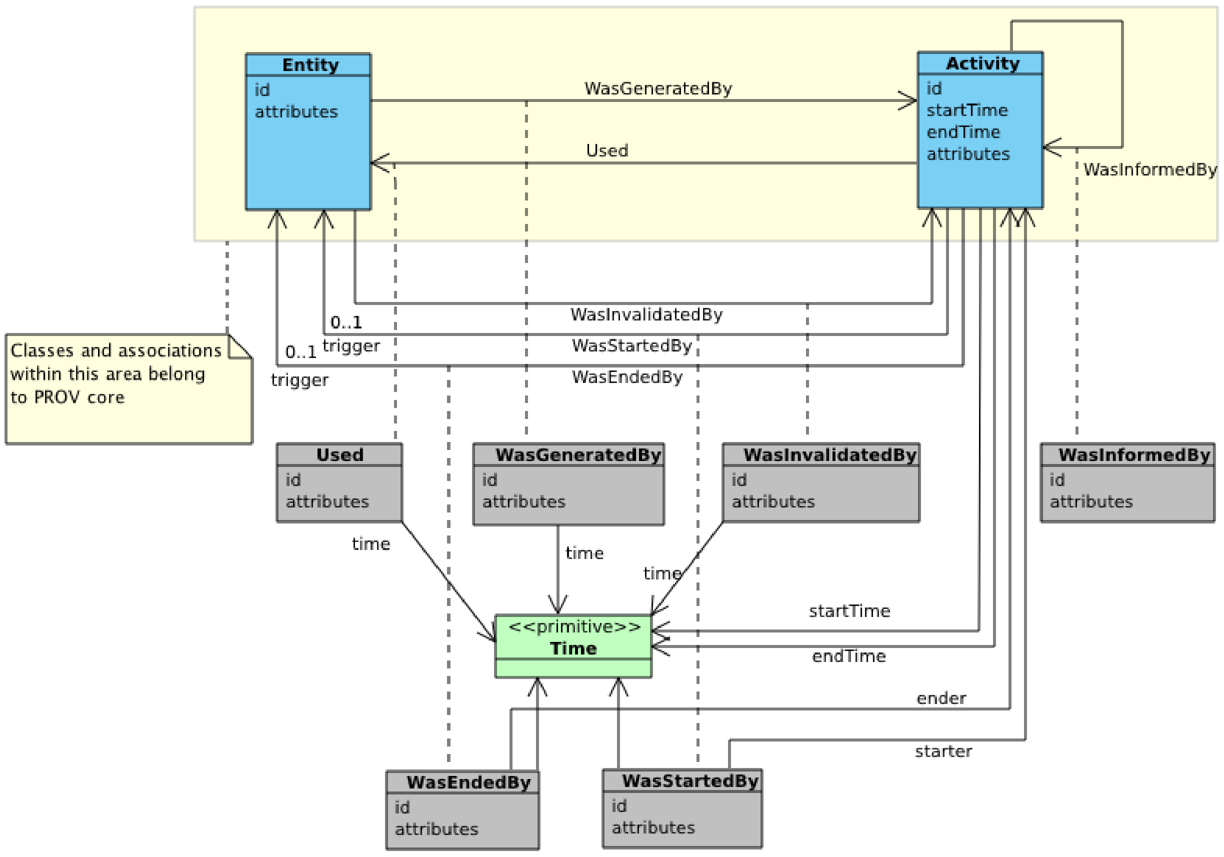

Figure 5◊: Entities and Activities Component Overview (Informative)

5.1.1 Entity

*An **entity**◊ is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.*

An ***entity***◊, written `entity(id, [attr1=val1, ...])` in PROV-N, has:

- *id*: an identifier for an entity;
- *attributes*: an OPTIONAL set of attribute-value pairs ((`attr1`, `val1`), ...) representing additional information about the fixed aspects of this entity.

Example 17◊

The following expression

```
entity(tr:WD-prov-dm-20111215, [ prov:type="document", ex:version="2" ])
```

states the existence of an entity, denoted by identifier `tr:WD-prov-dm-20111215`, with type `document` and version number `2`. The attribute `ex:version` is application specific, whereas the attribute `type` (see Section 5.7.4.4) is reserved in the PROV namespace.

5.1.2 Activity

*An **activity**◊ is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.*

An ***activity***◊, written `activity(id, st, et, [attr1=val1, ...])` in PROV-N, has:

- *id*: an identifier for an activity;
- *startTime*: an OPTIONAL time (`st`) for the start of the activity;
- *endTime*: an OPTIONAL time (`et`) for the end of the activity;
- *attributes*: an OPTIONAL set of attribute-value pairs ((`attr1`, `val1`), ...) representing additional information about this activity.

Example 18◊

The following expression

```
activity(a1, 2011-11-16T16:05:00, 2011-11-16T16:06:00,
        [ ex:host="server.example.org", prov:type='ex:edit' ])
```

states the existence of an activity with identifier `a1`, start time `2011-11-16T16:05:00`, and end time `2011-11-16T16:06:00`, running on host `server.example.org`, and of type `edit`. The attribute `host` is application specific (declared in some namespace with prefix `ex`). The attribute `type` is a reserved attribute of PROV-DM, allowing for sub-typing to be expressed (see Section 5.7.2.4).

Further considerations:

- An activity is not an entity. This distinction is similar to the distinction between 'continuant' and 'occurrent' in logic [Logic].

5.1.3 Generation

Generation ◊ *is the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation.*

Given that a generation is the completion of production of an entity, it is instantaneous.

Generation ◊, written `wasGeneratedBy(id; e, a, t, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for a generation;
- *entity*: an identifier (`e`) for a created entity;
- *activity*: an OPTIONAL identifier (`a`) for the activity that creates the entity;
- *time*: an OPTIONAL "generation time" (`t`), the time at which the entity was completely created;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this generation.

While each of *id*, *activity*, *time*, and *attributes* is OPTIONAL, at least one of them MUST be present.

Example 19 ◊

The following expressions

```
wasGeneratedBy(e1, a1, 2001-10-26T21:32:52, [ ex:port="p1" ])
wasGeneratedBy(e2, a1, 2001-10-26T10:00:00, [ ex:port="p2" ])
```

state the existence of two generations (with respective times `2001-10-26T21:32:52` and `2001-10-26T10:00:00`), at which new entities, identified by `e1` and `e2`, were created by an activity, identified by `a1`. The first one was available on port `p1`, whereas the other was available on port `p2`. The semantics of `port` are application specific.

Example 20 ◊

In some cases, we may want to record the time at which an entity was generated without having to specify the activity that generated it. To support this requirement, the activity element in generation is optional. Hence, the following expression indicates the time at which an entity is generated, without naming the activity that did it.

```
wasGeneratedBy(e, -, 2001-10-26T21:32:52)
```

5.1.4 Usage

Usage ◊ *is the beginning of utilizing an entity by an activity. Before usage, the activity had not begun to utilize this entity and could not have been affected by the entity.* (Note: This definition is formulated for a given usage; it is permitted for an activity to have used a same entity multiple times.)

Given that a usage is the beginning of utilizing an entity, it is instantaneous.

Usage ◊, written `used(id; a, e, t, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for a usage;
- *activity*: an identifier (`a`) for the activity that used an entity;
- *entity*: an OPTIONAL identifier (`e`) for the entity being used;
- *time*: an OPTIONAL "usage time" (`t`), the time at which the entity started to be used;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this usage.

While each of *id*, *entity*, *time*, and *attributes* is OPTIONAL, at least one of them MUST be present.

A reference to a given entity MAY appear in multiple usages that share a given activity identifier.

Example 21◊

The following usages

```
    used(a1, e1, 2011-11-16T16:00:00, [ ex:parameter="p1" ])
    used(a1, e2, 2011-11-16T16:00:01, [ ex:parameter="p2" ])
```

state that the activity identified by `a1` used two entities identified by `e1` and `e2`, at times `2011-11-16T16:00:00` and `2011-11-16T16:00:01`, respectively; the first one was found as the value of parameter `p1`, whereas the second was found as value of parameter `p2`. The semantics of `parameter` is application specific.

5.1.5 Communication

Communication◊ *is the exchange of some unspecified entity by two activities, one activity using some entity generated by the other.*

A communication implies that activity `a2` is dependent on another `a1`, by way of some unspecified entity that is generated by `a1` and used by `a2`.

A ***communication***◊, written as `wasInformedBy(id; a2, a1, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier identifying the relation;
- *informed*: the identifier (`a2`) of the informed activity;
- *informant*: the identifier (`a1`) of the informant activity;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this communication.

Example 22◊

Consider two activities `a1` and `a2`, the former performed by a government agency, and the latter by a driver caught speeding.

```
  activity(a1, [ prov:type="traffic regulations enforcing" ])
  activity(a2, [ prov:type="fine paying" ])
  wasInformedBy(a2, a1)
```

The last line indicates that some implicit entity was generated by `a1` and used by `a2`; this entity may be a traffic ticket that had a notice of fine, amount, and payment mailing details.

5.1.6 Start

Start◊ *is when an activity is deemed to have been started by an entity, known as* ***trigger***◊*. The activity did not exist before its start. Any usage, generation, or invalidation involving an activity follows the activity's start. A start may refer to a trigger entity that set off the activity, or to an activity, known as* ***starter***◊*, that generated the trigger.*

Given that a start is when an activity is deemed to have started, it is instantaneous.

An activity ***start***◊, written `wasStartedBy(id; a2, e, a1, t, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for the activity start;
- *activity*: an identifier (`a2`) for the started activity;
- *trigger*: an OPTIONAL identifier (`e`) for the entity triggering the activity;
- *starter*: an OPTIONAL identifier (`a1`) for the activity that generated the (possibly unspecified) entity (`e`);
- *time*: the OPTIONAL time (`t`) at which the activity was started;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this activity start.

While each of *id*, *trigger*, *starter*, *time*, and *attributes* is OPTIONAL, at least one of them MUST be present.

Example 23◊

The following example contains the description of an activity `a1` (a discussion), which was started at a specific time, and was triggered by an email message `e1`.

```
  entity(e1, [ prov:type="email message"] )
  activity(a1, [ prov:type="Discuss" ])
  wasStartedBy(a1, e1, -, 2011-11-16T16:05:00)
```

Furthermore, if the message is also an input to the activity, this can be described as follows:

```
used(a1, e1, -)
```

Alternatively, one can also describe the activity that generated the email message.

```
activity(a0, [ prov:type="Write" ])
wasGeneratedBy(e1, a0)
wasStartedBy(a1, e1, a0, 2011-11-16T16:05:00)
```

If `e1` is not known, it would also be valid to write:

```
wasStartedBy(a1, -, a0, 2011-11-16T16:05:00)
```

Example 24◊

In the following example, a race is started by a bang, and responsibility for this trigger is attributed to an agent `ex:Bob`.

```
activity(ex:foot_race)
entity(ex:bang)
wasStartedBy(ex:foot_race, ex:bang, -, 2012-03-09T08:05:08-05:00)
agent(ex:Bob)
wasAttributedTo(ex:bang, ex:Bob)
```

Example 25◊

In this example, filling the fuel tank was started as a consequence of observing low fuel. The trigger entity is unspecified, it could for instance have been the low fuel warning light, the fuel tank indicator needle position, or the engine not running properly.

```
activity(ex:filling-fuel)
activity(ex:observing-low-fuel)

agent(ex:driver, [ prov:type='prov:Person'  )
wasAssociatedWith(ex:filling-fuel, ex:driver)
wasAssociatedWith(ex:observing-low-fuel, ex:driver)

wasStartedBy(ex:filling-fuel, -, ex:observing-low-fuel, -)
```

The relations wasStartedBy and used are orthogonal, and thus need to be expressed independently, according to the situation being described.

5.1.7 End

End◊ *is when an activity is deemed to have been ended by an entity, known as* ***trigger***◊*. The activity no longer exists after its end. Any usage, generation, or invalidation involving an activity precedes the activity's end. An end may refer to a trigger entity that terminated the activity, or to an activity, known as* ***ender***◊ *that generated the trigger.*

Given that an end is when an activity is deemed to have ended, it is instantaneous.

An activity ***end***◊, written `wasEndedBy(id; a2, e, a1, t, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for the activity end;
- *activity*: an identifier (`a2`) for the ended activity;
- *trigger*: an OPTIONAL identifier (`e`) for the entity triggering the activity ending;
- *ender*: an OPTIONAL identifier (`a1`) for the activity that generated the (possibly unspecified) entity (`e`);
- *time*: the OPTIONAL time (`t`) at which the activity was ended;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this activity end.

While each of *id*, *trigger*, *ender*, *time*, and *attributes* is OPTIONAL, at least one of them MUST be present.

Example 26◊

The following example is a description of an activity `a1` (editing) that was ended following an approval document `e1`.

```
entity(e1, [ prov:type="approval document" ])
activity(a1, [ prov:type="Editing" ])
wasEndedBy(a1, e1, -, -)
```

5.1.8 Invalidation

Invalidation◊ *is the start of the destruction, cessation, or expiry of an existing entity by an activity. The entity is no longer available for use (or further invalidation) after invalidation. Any generation or usage of an entity precedes its invalidation.*

Given that an invalidation is the start of destruction, cessation, or expiry, it is instantaneous.

Entities have a duration. Generation marks the beginning of an entity, whereas invalidation marks its end. An entity's lifetime can end for different reasons:

- an entity was destroyed: e.g. a painting was destroyed by fire; a Web page is taken out of a site;
- an entity was consumed: e.g. Bob ate all his soup, Alice ran out of gas when driving to work;
- an entity expires: e.g. a "buy one beer, get one free" offer is valid during happy hour (7-8pm);
- an entity is time limited: e.g. the BBC news site on April 3rd, 2012;
- an entity attribute is changing: e.g. the traffic light changed from green to red.

In the first two cases, the entity has physically disappeared after its termination: there is no more soup, or painting. In the third case, there may be an "offer voucher" that still exists, but it is no longer valid; likewise, on April 4th, the BBC news site still exists but it is not the same entity as BBC news Web site on April 3rd; or the green traffic light (an entity with a fixed aspect green light) became the red traffic light (another entity with a fixed aspect red light).

Invalidation◊, written `wasInvalidatedBy(id; e, a, t, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for a invalidation;
- *entity*: an identifier for the invalidated entity;
- *activity*: an OPTIONAL identifier for the activity that invalidated the entity;
- *time*: an OPTIONAL "invalidation time", the time at which the entity began to be invalidated;
- *attributes*: an OPTIONAL set of attribute-value pairs representing additional information about this invalidation.

While each of *id*, *activity*, *time*, and *attributes* is OPTIONAL, at least one of them MUST be present.

Example 27◊

The Painter, a Picasso painting, is known to have been destroyed in a plane accident.

```
entity(ex:The-Painter)
agent(ex:Picasso)
wasAttributedTo(ex:The-Painter, ex:Picasso)
activity(ex:crash)
wasInvalidatedBy(ex:The-Painter, ex:crash, 1998-09-03T01:31:00, [ ex:circumstances="plane accident" ])
```

Example 28◊

The BBC news home page on 2012-04-03 `ex:bbcNews2012-04-03` contained a reference to a given news item bbc:news/uk-17595024, but the BBC news home page on the next day did not.

```
entity(ex:bbcNews2012-04-03)
hadMember(ex:bbcNews2012-04-03, bbc:news/uk-17595024)
wasGeneratedBy  (ex:bbcNews2012-04-03, -, 2012-04-03T00:00:01)
wasInvalidatedBy(ex:bbcNews2012-04-03, -, 2012-04-03T23:59:59)
```

We refer to example Example 43 for further descriptions of the BBC Web site, and to Section 5.6.2 for a description of the relation hadMember.

Example 29◊

In this example, the "buy one beer, get one free" offer expired at the end of the happy hour.

```
entity(buy_one_beer_get_one_free_offer_during_happy_hour)
wasAttributedTo(buy_one_beer_get_one_free_offer_during_happy_hour, proprietor)
wasInvalidatedBy(buy_one_beer_get_one_free_offer_during_happy_hour,
                 -,2012-03-10T18:00:00)
```

In contrast, in the following descriptions, Bob redeemed the offer 45 minutes before it expired, and got two beers.

```
entity(buy_one_beer_get_one_free_offer_during_happy_hour)
wasAttributedTo(buy_one_beer_get_one_free_offer_during_happy_hour, proprietor)
activity(redeemOffer)
entity(twoBeers)

wasAssociatedWith(redeemOffer, bob)
used(redeemOffer,
     buy_one_beer_get_one_free_offer_during_happy_hour,
     2012-03-10T17:15:00)
wasInvalidatedBy(buy_one_beer_get_one_free_offer_during_happy_hour,
                 redeemOffer,
                 2012-03-10T17:15:00)
wasGeneratedBy(twoBeers,redeemOffer)
```

We see that the offer was both used to be converted into `twoBeers` and invalidated by the `redeemOffer` activity: in other words, the combined usage and invalidation indicate consumption of the offer.

5.2 Component 2: Derivations

The second component of PROV-DM is concerned with: derivations of entities from other entities and derivation subtypes WasRevisionOf (Revision), WasQuotedFrom (Quotation), and HasPrimarySource (Primary Source). Figure 6 depicts the third component with PROV core structures in the yellow area, including two classes (Entity, Activity) and binary association WasDerivedFrom (Derivation). PROV extended structures are found outside this area. UML association classes express expanded n-ary relations. The subclasses are marked by the UML stereotype "prov:type" to indicate that the corresponding types are valid values for the attribute prov:type.

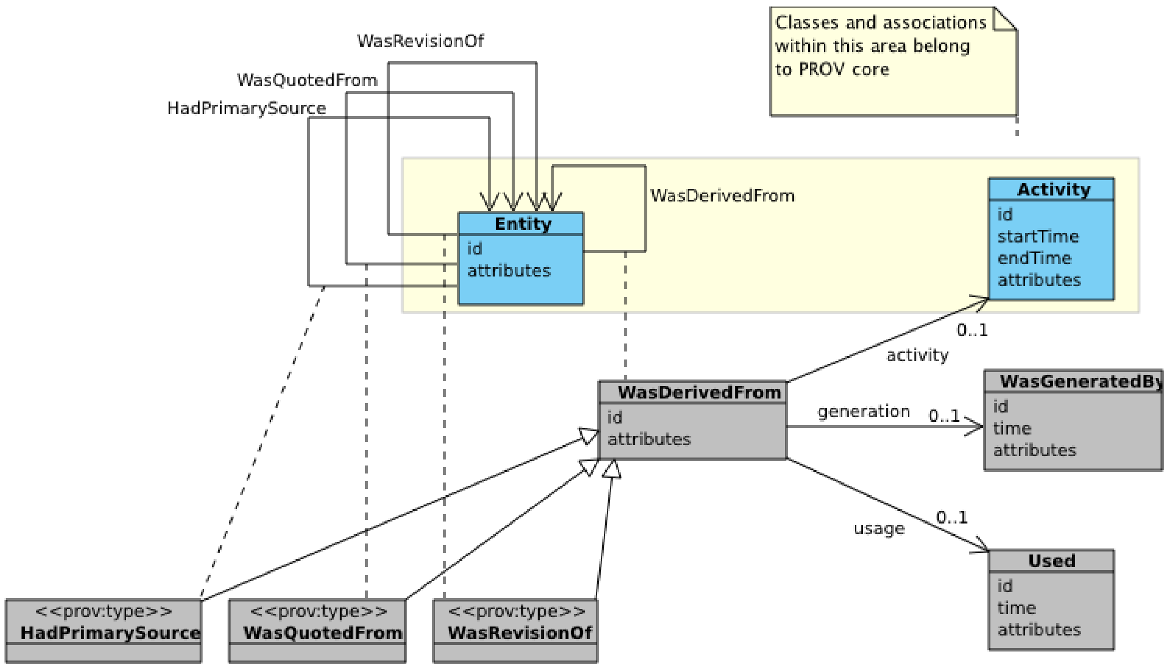

Figure 6: Derivation Component Overview (Informative)

5.2.1 Derivation

A ***derivation*** *is a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity.*

According to Section 2, for an entity to be transformed from, created from, or resulting from an update to another, there must be some underpinning activity or activities performing the necessary action(s) resulting in such a derivation. A derivation can be described at various levels of precision. In its simplest form, derivation relates two entities. Optionally, attributes can be added to represent further information about the derivation. If the derivation is the result of a single known activity, then this activity can also be optionally expressed. To provide a completely accurate description of the derivation, the generation and usage of the generated and used entities, respectively, can be provided, so as to make the derivation path, through usage, activity, and generation, explicit. Optional information such as activity, generation, and usage can be linked to derivations to aid analysis of provenance and to facilitate provenance-based reproducibility.

A ***derivation***, written `wasDerivedFrom(id; e2, e1, a, g2, u1, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for a derivation;
- *generatedEntity*: the identifier (`e2`) of the entity generated by the derivation;
- *usedEntity*: the identifier (`e1`) of the entity used by the derivation;
- *activity*: an OPTIONAL identifier (`a`) for the activity using and generating the above entities;
- *generation*: an OPTIONAL identifier (`g2`) for the generation involving the generated entity (`e2`) and activity (`a`);
- *usage*: an OPTIONAL identifier (`u1`) for the usage involving the used entity (`e1`) and activity (`a`);
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this derivation.

Example 30

The following descriptions are about derivations between `e2` and `e1`, but no information is provided as to the identity of the activity (and usage and generation) underpinning the derivation. In the second line, a type attribute is also provided.

```
    wasDerivedFrom(e2, e1)
    wasDerivedFrom(e2, e1, [ prov:type="physical transform" ])
```

The following description expresses that activity `a`, using the entity `e1` according to usage `u1`, derived the entity `e2` and generated it according to generation `g2`. It is followed by descriptions for generation `g2` and usage `u1`.

```
    wasDerivedFrom(e2, e1, a, g2, u1)
    wasGeneratedBy(g2; e2, a, -)
    used(u1; a, e1, -)
```

With such a comprehensive description of derivation, a program that analyzes provenance can identify the activity underpinning the derivation, it can identify how the preceding entity `e1` was used by the activity (e.g. for instance, which argument it was passed as, if the activity is the result of a function invocation), and which output the derived entity `e2` was obtained from (say, for a function returning multiple results).

5.2.2 Revision

*A **revision**◊ is a derivation for which the resulting entity is a revised version of some original.*

The implication here is that the resulting entity contains substantial content from the original. A ***revision***◊ relation is a kind of derivation relation from a revised entity to a preceding entity. The type of a revision relation is denoted by: ***`prov:Revision`***◊. PROV defines no revision-specific attributes.

Example 31◊

Revisiting the example of Section 4.2, we can now state that the report `tr:WD-prov-dm-20111215` was a revision of the report `tr:WD-prov-dm-20111018`.

```
    entity(tr:WD-prov-dm-20111215, [ prov:type='rec54:WD'  ])
    entity(tr:WD-prov-dm-20111018, [ prov:type='rec54:WD'  ])
    wasDerivedFrom(tr:WD-prov-dm-20111215,
                   tr:WD-prov-dm-20111018,
                   [ prov:type='prov:Revision' ])
```

5.2.3 Quotation

*A **quotation**◊ is the repeat of (some or all of) an entity, such as text or image, by someone who may or may not be its original author.*

A ***quotation***◊ relation is a kind of derivation relation, for which an entity was derived from a preceding entity by copying, or "quoting", some or all of it. The type of a quotation relation is denoted by: ***`prov:Quotation`***◊. PROV defines no quotation-specific attributes.

Example 32◊

The following paragraph is a quote from one of the author's blogs.

> *"During the workshop, it became clear to me that the consensus based models (which are often graphical in nature) can not only be formalized but also be directly connected to these database focused formalizations. I just needed to get over the differences in syntax. This could imply that we could have nice way to trace provenance across systems and through databases and be able to understand the mathematical properties of this interconnection."*

If `wp:thoughts-from-the-dagstuhl-principles-of-provenance-workshop/` denotes the original blog by agent `ex:Paul`, and `dm:bl-dagstuhl` denotes the above paragraph, then the following descriptions express that the above paragraph was copied by agent `ex:Luc` from a part of the blog, attributed to the agent `ex:Paul`.

```
    entity(wp:thoughts-from-the-dagstuhl-principles-of-provenance-workshop/)
    entity(dm:bl-dagstuhl)
    agent(ex:Luc)
    agent(ex:Paul)
    wasDerivedFrom(dm:bl-dagstuhl,
                   wp:thoughts-from-the-dagstuhl-principles-of-provenance-workshop/,
                   [ prov:type='prov:Quotation' ])
    wasAttributedTo(dm:bl-dagstuhl, ex:Luc)
    wasAttributedTo(wp:thoughts-from-the-dagstuhl-principles-of-provenance-workshop/, ex:Paul)
```

5.2.4 Primary Source

*A **primary source**◊ for a topic refers to something produced by some agent with direct experience and knowledge about the topic, at the time of the topic's study, without benefit from hindsight.*

Because of the directness of primary sources, they "speak for themselves" in ways that cannot be captured through the filter of secondary sources. As such, it is important for secondary sources to reference those primary sources from which they were derived, so that their reliability can be investigated.

It is also important to note that a given entity might be a primary source for one entity but not another. It is the reason why Primary Source is defined as a relation as opposed to a subtype of Entity.

A ***primary source***◊ relation is a kind of a derivation relation from secondary materials to their primary sources. It is recognized that the determination of primary sources can be up to interpretation, and should be done according to conventions accepted within the application's domain. The type of a primary source relation is denoted by: ***`prov:PrimarySource`***◊. PROV defines no attributes specific to primary source.

Example 33◊

Let us consider Charles Joseph Minard's flow map of Napoleon's March in 1812, which was published in 1869. Although the map is not a primary source, Minard probably used the journal of Pierre-Irénée Jacob, pharmacist to Napoleon's army during the Russian campaign. This primary source relation can be encoded as follows.

```
entity(ex:la-campagne-de-Russie-1812-1813, [ prov:type="map" ])
entity(ex:revue-d-Histoire-de-la-Pharmacie-t-XVIII, [ prov:type="journal" ])
wasDerivedFrom(ex:la-campagne-de-Russie-1812-1813,
               ex:revue-d-Histoire-de-la-Pharmacie-t-XVIII,
               [ prov:type='prov:PrimarySource' ])
```

5.3 Component 3: Agents, Responsibility, and Influence

The third component of PROV-DM, depicted in Figure 7, is concerned with agents and the relations WasAttributedTo (Attribution), WasAssociatedWith (Association), and ActedOnBehalfOf (Delegation), relating agents to entities, activities, and agents, respectively. Core structures are displayed in the yellow area and include three classes and three binary associations. Outside the yellow area, extended structures comprise UML association classes to express expanded n-ary relations, and subclasses Plan, Person, SoftwareAgent, and Organization. The subclasses are marked by the UML stereotype "prov:type" to indicate that that these are valid values for the attribute prov:type.

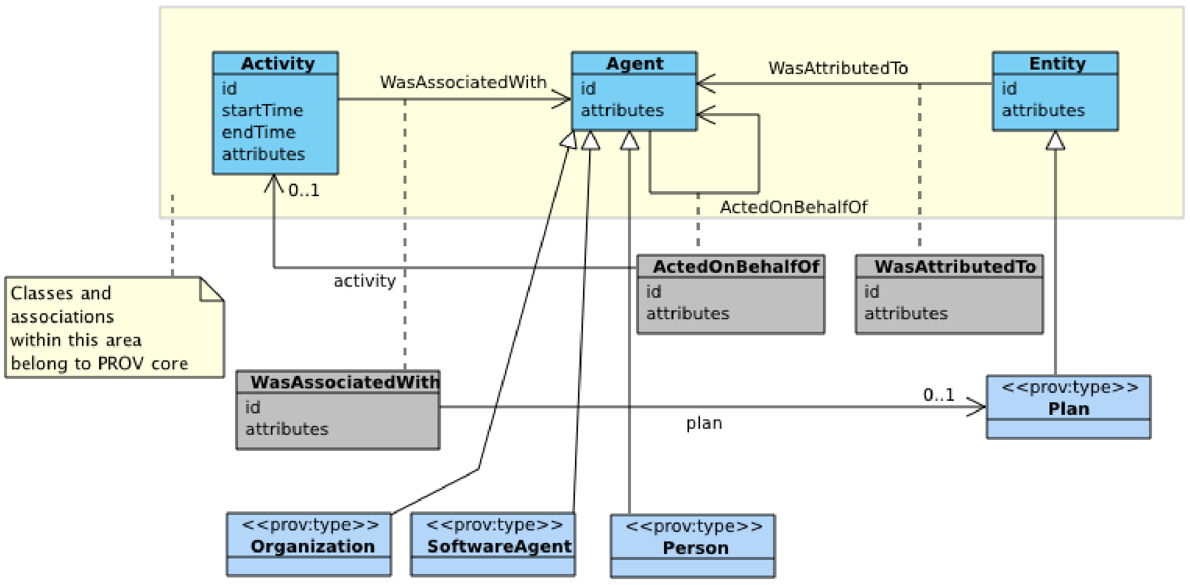

Figure 7◊: Agents and Responsibility Overview (Informative)

Component 3 further defines a general notion of influence, a relation implied by all relations defined so far. Figure 8 displays one new association class, generalizing previously introduced associations.

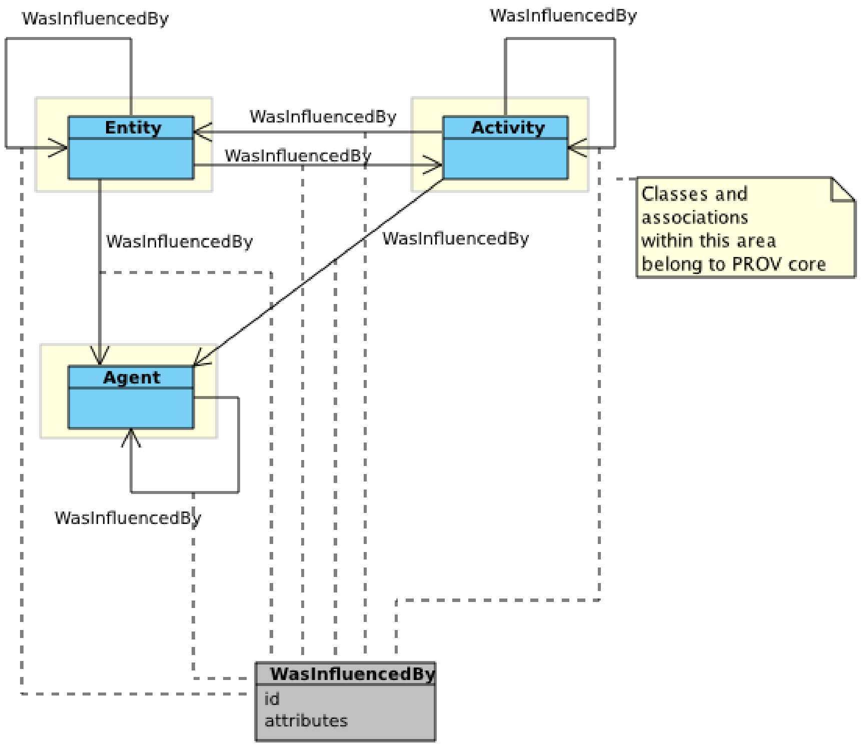

Figure 8: Influence Overview (Informative)

5.3.1 Agent

*An **agent** is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.*

An agent may be a particular type of entity or activity. This means that the model can be used to express provenance of the agents themselves.

An ***agent***, written `agent(id, [attr1=val1, ...])` in PROV-N, has:

- *id*: an identifier for an agent;
- *attributes*: a set of attribute-value pairs ((`attr1`, `val1`), ...) representing additional information about this agent.

It is useful to define some basic categories of agents from an interoperability perspective. There are three types of agents that are common across most anticipated domains of use; it is acknowledged that these types do not cover all kinds of agent.

- `SoftwareAgent`

  *A **software agent** is running software.* The type of a software agent is denoted by ***`prov:SoftwareAgent`***.

- `Organization`

  *An **organization** is a social or legal institution such as a company, society, etc.* The type of an organization agent is denoted by ***`prov:Organization`***.

- `Person`

  Person** agents are people.* The type of a person agent is denoted by ***`prov:Person`.

PROV defines no attributes specific to SoftwareAgent, Organization, and Person.

Example 34

The following expression is about an agent identified by `e1`, which is a person, named Alice, with employee number 1234.

```
agent(e1, [ex:employee="1234", ex:name="Alice", prov:type='prov:Person' ])
```

It is optional to specify the type of an agent. When present, it is expressed using the `prov:type` attribute.

5.3.2 Attribution

Attribution◊ *is the ascribing of an entity to an agent.*

When an entity `e` is attributed to agent `ag`, entity `e` was generated by some unspecified activity that in turn was associated to agent `ag`. Thus, this relation is useful when the activity is not known, or irrelevant.

An ***attribution***◊ relation, written `wasAttributedTo(id; e, ag, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for the relation;
- *entity*: an entity identifier (`e`);
- *agent*: the identifier (`ag`) of the agent whom the entity is ascribed to, and therefore bears some responsibility for its existence;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this attribution.

Example 35◊

Revisiting the example of Section 4.1, we can ascribe `tr:WD-prov-dm-20111215` to some agents without an explicit activity.

```
agent(ex:Paolo, [ prov:type='prov:Person' ])
agent(ex:Simon, [ prov:type='prov:Person' ])
entity(tr:WD-prov-dm-20111215, [ prov:type='rec54:WD' ])
wasAttributedTo(tr:WD-prov-dm-20111215, ex:Paolo, [ prov:type="editorship" ])
wasAttributedTo(tr:WD-prov-dm-20111215, ex:Simon, [ prov:type="authorship" ])
```

5.3.3 Association

An activity ***association***◊ *is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity. It further allows for a plan to be specified, which is the plan intended by the agent to achieve some goals in the context of this activity.*

An ***association***◊, written `wasAssociatedWith(id; a, ag, pl, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for the association between an activity and an agent;
- *activity*: an identifier (`a`) for the activity;
- *agent*: an OPTIONAL identifier (`ag`) for the agent associated with the activity;
- *plan*: an OPTIONAL identifier (`pl`) for the plan the agent relied on in the context of this activity;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this association of this activity with this agent.

While each of *id*, *agent*, *plan*, and *attributes* is OPTIONAL, at least one of them MUST be present.

A ***plan***◊ *is an entity that represents a set of actions or steps intended by one or more agents to achieve some goals.* The type of a Plan entity is denoted by ***`prov:Plan`***◊.

PROV defines no plan-specific attributes.

Example 36◊

In the following example, a designer agent and an operator agent are associated with an activity. The designer's goals are achieved by a workflow `ex:wf`, described as an entity of type `plan`.

```
activity(ex:a, [ prov:type="workflow execution" ])
agent(ex:ag1,  [ prov:type="operator" ])
agent(ex:ag2,  [ prov:type="designer" ])
wasAssociatedWith(ex:a, ex:ag1, -,     [ prov:role="loggedInUser", ex:how="webapp" ])
wasAssociatedWith(ex:a, ex:ag2, ex:wf, [ prov:role="designer", ex:context="project1" ])
entity(ex:wf, [ prov:type='prov:Plan' ,
                ex:label="Workflow 1",
                prov:location="http://example.org/workflow1.bpel" %% xsd:anyURI ])
```

Since the workflow `ex:wf` is itself an entity, its provenance can also be expressed in PROV: it can be generated by some activity and derived from other entities, for instance.

Example 37◊

In some cases, one wants to indicate a plan was followed, without having to specify which agent was involved.

```
activity(ex:a, [ prov:type="workflow execution" ])
```

```
    wasAssociatedWith(ex:a, -, ex:wf)
    entity(ex:wf, [ prov:type='prov:Plan',
                    ex:label="Workflow 1",
                    ex:url="http://example.org/workflow1.bpel" %% xsd:anyURI])
```

In this case, it is assumed that an agent exists, but it has not been specified.

5.3.4 Delegation

Delegation◊ *is the assignment of authority and responsibility to an agent (by itself or by another agent) to carry out a specific activity as a delegate or representative, while the agent it acts on behalf of retains some responsibility for the outcome of the delegated work.*

For example, a student acted on behalf of his or her supervisor, who acted on behalf of the department chair, who acted on behalf of the university; all those agents are responsible in some way for the activity that took place but we do not say explicitly who bears responsibility and to what degree.

A ***delegation***◊ link, written `actedOnBehalfOf(id; ag2, ag1, a, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier for the delegation link between delegate and responsible;
- *delegate*: an identifier (`ag2`) for the agent associated with an activity, acting on behalf of the responsible agent;
- *responsible*: an identifier (`ag1`) for the agent, on behalf of which the delegate agent acted;
- *activity*: an OPTIONAL identifier (`a`) of an activity for which the delegation link holds;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this delegation link.

Example 38◊

The following fragment describes three agents: a programmer, a researcher, and a funder. The programmer and researcher are associated with a workflow activity. The programmer acts on behalf of the researcher (line-management) encoding the commands specified by the researcher; the researcher acts on behalf of the funder, who has a contractual agreement with the researcher. The terms 'line-management' and 'contract' used in this example are domain specific.

```
    activity(a,[ prov:type="workflow" ])
    agent(ag1, [ prov:type="programmer" ])
    agent(ag2, [ prov:type="researcher" ])
    agent(ag3, [ prov:type="funder" ])
    wasAssociatedWith(a, ag1, [ prov:role="loggedInUser" ])
    wasAssociatedWith(a, ag2)
    wasAssociatedWith(a, ag3)
    actedOnBehalfOf(ag1, ag2, a, [ prov:type="line-management" ])
    actedOnBehalfOf(ag2, ag3, a, [ prov:type="contract" ])
```

5.3.5 Influence

Influence◊ *is the capacity of an entity, activity, or agent to have an effect on the character, development, or behavior of another by means of usage, start, end, generation, invalidation, communication, derivation, attribution, association, or delegation.*

An influence relation between two objects `o2` and `o1` is a generic dependency of `o2` on `o1` that signifies some form of influence of `o1` on `o2`.

An ***Influence***◊ relation, written `wasInfluencedBy(id; o2, o1, attrs)` in PROV-N, has:

- *id*: an OPTIONAL identifier identifying the relation;
- *influencee*: an identifier (`o2`) for an entity, activity, or agent;
- *influencer*: an identifier (`o1`) for an ancestor entity, activity, or agent that the former depends on;
- *attributes*: an OPTIONAL set (`attrs`) of attribute-value pairs representing additional information about this relation.

A usage, start, end, generation, invalidation, communication, derivation, attribution, association, and delegation is also an influence. It is RECOMMENDED to adopt these more specific relations when writing provenance descriptions. It is anticipated that the Influence relation may be useful to express queries over provenance information.

The following table establishes the correspondence between the attributes *influencee* and *influencer*, and attributes of Usage, Start, End, Generation, Invalidation, Communication, Derivation, Attribution, Association, and Delegation.

Table 7◊: Mapping Relations to Influence

Relation Name	*influencee*	*influencer*
Generation	*entity*	*activity*

Usage	*activity*	*entity*
Communication	*informed*	*informant*
Start	*activity*	*trigger*
End	*activity*	*trigger*
Invalidation	*entity*	*activity*
Derivation	*generatedEntity*	*usedEntity*
Attribution	*entity*	*agent*
Association	*activity*	*agent*
Delegation	*delegate*	*responsible*

Example 39◊

We refer to the example of Section 4.2, and specifically to Figure 3. We could have expressed that the influence of `w3:Consortium` on `tr:WD-prov-dm-20111215`.

```
    wasInfluencedBy(tr:WD-prov-dm-20111215, w3:Consortium)
```

Instead, it is recommended to express the more specific description:

```
    wasAttributedTo(tr:WD-prov-dm-20111215, w3:Consortium)
```

5.4 Component 4: Bundles

The fourth component of PROV-DM is concerned with bundles, a mechanism to support provenance of provenance. Figure 9 depicts a UML class diagram for the fourth component. It comprises a Bundle class defined as a subclass of Entity.

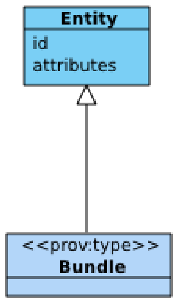

Figure 9◊: Bundle Component Overview (Informative)

5.4.1 Bundle constructor

*A **bundle**◊ is a named set of provenance descriptions, and is itself an entity, so allowing provenance of provenance to be expressed.*

A ***bundle constructor***◊ allows the content and the name of a bundle to be specified; it is written `bundle id description_1 ... description_n endBundle` and consists of:

- *id*: an identifier for the bundle;
- *descriptions*: a set of provenance descriptions `description_1`, ..., `description_n`.

A bundle's identifier `id` identifies a unique set of descriptions.

There may be other kinds of bundles not directly expressible by this constructor, such as provenance descriptions handwritten on a letter or a whiteboard, etc. Whatever the means by which bundles are expressed, all can be described, as in the following section.

5.4.2 Bundle Type

A bundle is a named set of descriptions, but it is also an entity so that its provenance can be described.

PROV defines the following ***type for bundles***◊:

- ***`prov:Bundle`***◊ is the type that denotes Bundle entities.

PROV defines no bundle-specific attributes.

A bundle description is of the form `entity(id, [ prov:type='prov:Bundle', attr1=val1, ...] )` where `id` is an identifier denoting a bundle, a type prov:Bundle and an OPTIONAL set of attribute-value pairs ((`attr1`, `val1`), ...) representing additional information about this bundle.

The provenance of provenance can then be described using PROV constructs, as illustrated by Example 40 and Example 41.

Example 40◊

Let us consider two entities `ex:report1` and `ex:report2`.

```
entity(ex:report1, [ prov:type="report", ex:version=1 ])
wasGeneratedBy(ex:report1, -, 2012-05-24T10:00:01)
entity(ex:report2, [ prov:type="report", ex:version=2])
wasGeneratedBy(ex:report2, -, 2012-05-25T11:00:01)
wasDerivedFrom(ex:report2, ex:report1)
```

Let us assume that Bob observed the creation of `ex:report1`. A first bundle can be expressed.

```
bundle bob:bundle1
  entity(ex:report1, [ prov:type="report", ex:version=1 ])
  wasGeneratedBy(ex:report1, -, 2012-05-24T10:00:01)
endBundle
```

In contrast, Alice observed the creation of `ex:report2` and its derivation from `ex:report1`. A separate bundle can also be expressed.

```
bundle alice:bundle2
  entity(ex:report1)
  entity(ex:report2, [ prov:type="report", ex:version=2 ])
  wasGeneratedBy(ex:report2, -, 2012-05-25T11:00:01)
  wasDerivedFrom(ex:report2, ex:report1)
endBundle
```

The first bundle contains the descriptions corresponding to Bob observing the creation of `ex:report1`. Its provenance can be described as follows.

```
entity(bob:bundle1, [ prov:type='prov:Bundle' ])
wasGeneratedBy(bob:bundle1, -, 2012-05-24T10:30:00)
wasAttributedTo(bob:bundle1, ex:Bob)
```

In contrast, the second bundle is attributed to Alice who observed the derivation of `ex:report2` from `ex:report1`.

```
entity(alice:bundle2, [ prov:type='prov:Bundle' ])
wasGeneratedBy(alice:bundle2, -, 2012-05-25T11:15:00)
wasAttributedTo(alice:bundle2, ex:Alice)
```

Example 41◊

A provenance aggregator could merge two bundles, resulting in a novel bundle, whose provenance is described as follows.

```
bundle agg:bundle3
  entity(ex:report1, [ prov:type="report", ex:version=1 ])
  wasGeneratedBy(ex:report1, -, 2012-05-24T10:00:01)

  entity(ex:report2, [ prov:type="report", ex:version=2 ])
  wasGeneratedBy(ex:report2, -, 2012-05-25T11:00:01)
  wasDerivedFrom(ex:report2, ex:report1)
endBundle

entity(agg:bundle3, [ prov:type='prov:Bundle' ])
agent(ex:aggregator01, [ prov:type='ex:Aggregator' ])
wasAttributedTo(agg:bundle3, ex:aggregator01)
wasDerivedFrom(agg:bundle3, bob:bundle1)
wasDerivedFrom(agg:bundle3, alice:bundle2)
```

The new bundle is given a new identifier `agg:bundle3` and is attributed to the `ex:aggregator01` agent.

5.5 Component 5: Alternate Entities

The fifth component of PROV-DM is concerned with relations SpecializationOf (Specialization) and AlternateOf (Alternate) between entities. Figure 10 depicts the fifth component with a single class and two binary associations.

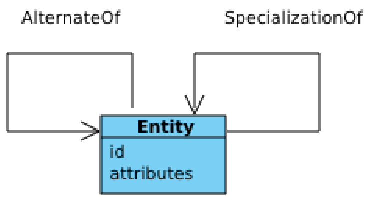

Figure 10◊: Alternates Component Overview (Informative)

Two provenance descriptions about the same thing may emphasize differents aspects of that thing.

Example 42◊

User Alice writes an article. In its provenance, she wishes to refer to the precise version of the article with a date-specific IRI, as she might edit the article later. Alternatively, user Bob refers to the article in general, independently of its variants over time.

The PROV data model introduces relations, called specialization and alternate that allow entities to be linked together. They are defined as follows.

5.5.1 Specialization

*An entity that is a **specialization**◊ of another shares all aspects of the latter, and additionally presents more specific aspects of the same thing as the latter. In particular, the lifetime of the entity being specialized contains that of any specialization.*

Examples of aspects include a time period, an abstraction, and a context associated with the entity.

A ***specialization***◊ relation, written `specializationOf(infra, supra)` in PROV-N, has:

- *specificEntity*: an identifier (`infra`) of the entity that is a specialization of the general entity (`supra`);
- *generalEntity*: an identifier (`supra`) of the entity that is being specialized.

A specialization is not, as defined here, also an influence, and therefore does not have an id and attributes.

Example 43◊

The BBC news home page on 2012-03-23 `ex:bbcNews2012-03-23` is a specialization of the BBC news page in general bbc:news/. This can be expressed as follows.

```
specializationOf(ex:bbcNews2012-03-23, bbc:news/)
```

We have created a new qualified name, `ex:bbcNews2012-03-23`, in the namespace `ex`, to identify the specific page carrying this day's news, which would otherwise be the generic `bbc:news/` page.

5.5.2 Alternate

*Two **alternate**◊ entities present aspects of the same thing. These aspects may be the same or different, and the alternate entities may or may not overlap in time.*

An ***alternate***◊ relation, written `alternateOf(e1, e2)` in PROV-N, has:

- *alternate1*: an identifier (`e1`) of the first of the two entities;
- *alternate2*: an identifier (`e2`) of the second of the two entities.

An alternate is not, as defined here, also an influence, and therefore does not have an id and attributes.

Note that alternateOf is a necessarily very general relationship that, in reasoning, only states that the two alternate entities respectively fix some aspects of some common thing (possibly evolving over time), and so there is some relevant connection between the provenance of the alternates. In a specific application context, alternateOf, or a subtype of it, could allow more inferences.

Example 44◊

A given news item on the BBC News site bbc:news/science-environment-17526723 for desktop is an alternate of a bbc:news/mobile/science-environment-17526723 for mobile devices.

```
entity(bbc:news/science-environment-17526723,
       [ prov:type="a news item for desktop"])
entity(bbc:news/mobile/science-environment-17526723,
       [ prov:type="a news item for mobile devices"])
alternateOf(bbc:news/science-environment-17526723,
            bbc:news/mobile/science-environment-17526723)
```

Example 45◊

Considering again the two versions of the technical report `tr:WD-prov-dm-20111215` (second working draft) and `tr:WD-prov-dm-20111018` (first working draft). They are alternates of each other.

```
entity(tr:WD-prov-dm-20111018)
entity(tr:WD-prov-dm-20111215)
alternateOf(tr:WD-prov-dm-20111018, tr:WD-prov-dm-20111215)
```

They are both specializations of the page http://www.w3.org/TR/prov-dm/.

5.6 Component 6: Collections

The sixth component of PROV-DM is concerned with the notion of collections. A collection is an entity that has some members. The members are themselves entities, and therefore their provenance can be expressed. Some applications need to be able to express the provenance of the collection itself: e.g. who maintains the collection (attribution), which members it contains as it evolves, and how it was assembled. The purpose of Component 6 is to define the types and relations that are useful to express the provenance of collections.

Figure 11 depicts the sixth component with two new classes (Collection, Empty Collection) and one association HadMember (Membership).

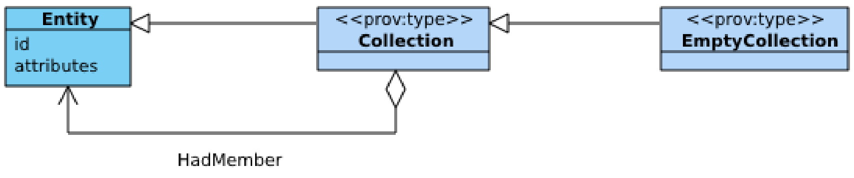

Figure 11◊: Collections Component Overview (Informative)

5.6.1 Collection

*A **collection**◊ is an entity that provides a structure to some constituents that must themselves be entities. These constituents are said to be **member**◊ of the collections. An **empty collection**◊ is a collection without members.*

PROV-DM defines the following types related to collections:

- ***`prov:Collection`***◊ denotes an entity of type Collection, i.e. an entity that can participate in relations amongst collections;
- ***`prov:EmptyCollection`***◊ denotes an empty collection.

PROV defines no collection-specific attributes.

Example 46◊

```
entity(c0, [ prov:type='prov:EmptyCollection' ])  // c0 is an empty collection
entity(c1, [ prov:type='prov:Collection'  ])      // c1 is a collection, with unknown content
```

5.6.2 Membership

A **membership** relation is defined for stating the members of a Collection.

Membership◊ *is the belonging of an entity to a collection.*

A ***membership***◊ relation, written `hadMember(c, e)`, has:

- *collection*: an identifier (`c`) for the collection whose member is asserted;
- *entity*: the identifier `e` of an entity that is member of the collection.

Membership is not, as defined here, also an influence, and therefore does not have an id and attributes.

Example 47◊

In this example, `c` is a collection known to have `e0`, `e1`, and `e2` as members, and may have other members.

```
entity(e0)
entity(e1)
entity(e2)

entity(c, [prov:type='prov:Collection'  ])      // c is a collection, with unknown content
hadMember(c, e0)
hadMember(c, e1)
hadMember(c, e2)
```

5.7 Further Elements of PROV-DM

This section introduces further elements of PROV-DM.

5.7.1 Identifier

An ***identifier***◊ is a qualified name.

Entity, Activity, and Agent have a mandatory identifier. Two entities (resp. activities, agents) are equal if they have the same identifier.

Generation, Usage, Communication, Start, End, Invalidation, Derivation, Attribution, Association, Delegation, Influence have an optional identifier. Two generations (resp. usages, communications, etc.) are equal if they have the same identifier.

5.7.2 Attribute

An ***attribute***◊ is a qualified name.

The PROV data model introduces a pre-defined set of attributes in the PROV namespace, which we define below. This specification does not provide any interpretation for any attribute declared in any other namespace.

Table 8◊: PROV-DM Attributes At a Glance

Attribute	Allowed In	value	Section
prov:label	*any construct*	A Value of type xsd:string	Section 5.7.2.1
prov:location	Entity, Activity, Agent, Usage, Generation, Invalidation, Start, and End	A Value	Section 5.7.2.2
prov:role	Usage, Generation, Invalidation, Association, Start, and End	A Value	Section 5.7.2.3
prov:type	*any construct*	A Value	Section 5.7.2.4
prov:value	Entity	A Value	Section 5.7.2.5

5.7.2.1 prov:label

*The attribute **`prov:label`**◊ provides a human-readable representation of an instance of a PROV-DM type or relation.* The value associated with the attribute `prov:label` MUST be a string.

Example 48◊

The following entity is provided with a label attribute.

```
entity(ex:e1, [ prov:label="This is a human-readable label" ])
```

The following entity has two label attributes, in French and English.

```
entity(ex:car01, [ prov:label="Voiture 01"@fr, prov:label="Car 01"@en ])
```

5.7.2.2 prov:location

*A **location**◊ can be an identifiable geographic place (ISO 19112), but it can also be a non-geographic place such as a directory, row, or column.* As such, there are numerous ways in which location can be expressed, such as by a coordinate, address, landmark, and so forth. This document does not specify how to concretely express locations, but instead provide a mechanism to introduce locations, by means of a reserved attribute.

The attribute `prov:location` is an OPTIONAL attribute of Entity, Activity, Agent, Usage, Generation, Invalidation, Start, and End. The value associated with the attribute `prov:location` MUST be a PROV-DM Value, expected to denote a location.

While the attribute `prov:location` is allowed for several PROV concepts, it may not make sense to use it in some cases. For example, an activity that describes the relocation of an entity will have start and end locations, as well as every place in between those points.

Example 49◊

The following expression describes entity Mona Lisa, a painting, with a location attribute.

```
entity(ex:MonaLisa, [ prov:location="Le Louvre, Paris", prov:type="StillImage" ])
```

The following expression describes a cell, at coordinates (5,5), with value 10.

```
entity(ex:cell, [ prov:location="(5,5)", prov:value="10" %% xsd:integer ])
```

5.7.2.3 prov:role

*A **role**◊ is the function of an entity or agent with respect to an activity, in the context of a usage, generation, invalidation, association, start, and end.*

The attribute `prov:role` is allowed to occur multiple times in a list of attribute-value pairs. The value associated with a `prov:role` attribute MUST be a PROV-DM Value.

Example 50◊

The following activity is associated with an agent acting as the operator.

```
wasAssociatedWith(a, ag, [ prov:role="operator" ])
```

In the following expression, the activity `ex:div01` used entity `ex:cell` in the role of divisor.

```
used(ex:div01, ex:cell, [ prov:role="divisor" ])
```

5.7.2.4 prov:type

*The attribute **`prov:type`**◊ provides further typing information for any construct with an optional set of attribute-value pairs.*

PROV-DM liberally defines a type as a category of things having common characteristics. PROV-DM is agnostic about the representation of types, and only states that the value associated with a `prov:type` attribute MUST be a PROV-DM Value. The attribute `prov:type` is allowed to occur multiple times.

Example 51◊

The following describes an agent of type software agent.

```
agent(ag, [ prov:type='prov:SoftwareAgent' ])
```

The following types are pre-defined in PROV, and are valid values for the `prov:type` attribute.

Table 9◊: PROV-DM Predefined Types

Type	Specification	Core concept
`prov:Bundle`	Section 5.4.1	Entity
`prov:Collection`	Section 5.6.1	Entity
`prov:EmptyCollection`	Section 5.6.1	Entity

prov:Organization	Section 5.3.1	Agent
prov:Person	Section 5.3.1	Agent
prov:Plan	Section 5.3.3	Entity
prov:PrimarySource	Section 5.2.4	Derivation
prov:Quotation	Section 5.2.3	Derivation
prov:Revision	Section 5.2.2	Derivation
prov:SoftwareAgent	Section 5.3.1	Agent

5.7.2.5 prov:value

*The attribute **prov:value**◊ provides a value that is a direct representation of an entity as a PROV-DM Value.*

The attribute prov:value is an OPTIONAL attribute of entity. The value associated with the attribute prov:value MUST be a PROV-DM Value. The attribute prov:value MAY occur at most once in a set of attribute-value pairs.

Example 52◊

The following example illustrates the provenance of the number 4 obtained by an activity that computed the length of an input string "abcd". The input and the output are expressed as entities ex:in and ex:out, respectively. They each have a prov:value attribute associated with the corresponding value.

```
entity(ex:in, [ prov:value="abcd" ])
entity(ex:out, [ prov:value=4 ])
activity(ex:len, [ prov:type="string-length" ])
used(ex:len, ex:in)
wasGeneratedBy(ex:out, ex:len)
wasDerivedFrom(ex:out, ex:in)
```

Two different entities MAY have the same value for the attribute prov:value. For instance, when two entities, with the same prov:value, are generated by two different activities, as illustrated by the following example.

Example 53◊

Example 52 illustrates an entity with a given value 4. This examples shows that another entity with the same value may be computed differently (by an addition).

```
entity(ex:in1, [ prov:value=3 ])
entity(ex:in2, [ prov:value=1 ])
entity(ex:out2, [ prov:value=4 ])      // ex:out2 also has value 4
activity(ex:add1, [ prov:type="addition" ])
used(ex:add1, ex:in1)
used(ex:add1, ex:in2)
wasGeneratedBy(ex:out2, ex:add1)
```

5.7.3 Value

*A **value**◊ is a constant such as a string, number, time, qualified name, IRI, and encoded binary data, whose interpretation is outside the scope of PROV.* Values can occur in attribute-value pairs.

Each kind of such values is called a *datatype*. Use of the following data types is RECOMMENDED.

- The RDF-compatible [RDF-CONCEPTS] types, including those taken from the set of XML Schema Datatypes [XMLSCHEMA11-2];
- Qualified names introduced in this specification.

The normative definitions of these datatypes are provided by their respective specifications.

Conformance to RDF Datatypes As of the publication of this document, RDF 1.1 Concepts and Abstract Syntax [RDF-CONCEPTS11] is not yet a W3C Recommendation (see http://www.w3.org/TR/rdf11-concepts/ for the latest version). Both the Provenance Working Group and the RDF Working Group are confident that there will be only minor changes before it becomes a W3C Recommendation. In order to take advantage of the anticipated corrections and new features sooner, while also providing stability in case the specification does not advance as expected, conformance to PROV as it relates to RDF Datatypes is defined as follows:

- If RDF 1.1 Concepts and Abstract Syntax becomes a W3C Recommendation, all references in PROV to RDF Concepts and Abstract Syntax will be normative references to the 1.1 Recommendation.
- Until that time, references in PROV to RDF Concepts and Abstract Syntax features operate as follows:

- If RDF 1.0 defines the features, then the reference is normative to the 1.0 definition [RDF-CONCEPTS];
- otherwise, the feature is optional in PROV and the reference is informative only.

This "change in normative reference" is effective as of the publication of RDF 1.1 as a W3C Recommendation. However, W3C expects to publish a new edition of PROV once RDF 1.1 becomes a Recommendation to update the reference explicitly.

Example 54◊

The following examples respectively are the string "abc", the integer number 1, and the IRI "http://example.org/foo".

```
     "abc"
     "1" %% xsd:integer
     "http://example.org/foo" %% xsd:anyURI
```

The following example shows a value of type `prov:QUALIFIED_NAME` (see `prov:QUALIFIED_NAME` [PROV-N]). The prefix `ex` must be bound to a namespace declared in a namespace declaration.

```
     "ex:value" %% prov:QUALIFIED_NAME
```

Alternatively, the same value can be expressed using the following convenience notation.

```
     'ex:value'
```

We note that PROV ***time instants***◊ are defined according to xsd:dateTime [XMLSCHEMA11-2].

Example 55◊

In the following example, the generation time of entity `e1` is expressed according to xsd:dateTime [XMLSCHEMA11-2].

```
     wasGeneratedBy(e1,a1, 2001-10-26T21:32:52)
```

5.7.4 Namespace Declaration

A ***namespace***◊ is identified by an IRI [RFC3987]. In PROV-DM, attributes, identifiers, and values with qualified names as data type can be placed in a namespace using the mechanisms described in this specification.

A ***namespace declaration***◊ consists of a binding between a prefix and a namespace. Every qualified name with this prefix in the scope of this declaration refers to this namespace.

A ***default namespace declaration***◊ consists of a namespace. Every un-prefixed qualified name refers to default namespace declaration.

The ***PROV namespace***◊ is identified by the IRI http://www.w3.org/ns/prov#.

5.7.5 Qualified Name

*A **qualified name**◊ is a name subject to namespace interpretation. It consists of a namespace, denoted by an optional prefix, and a local name.*

PROV-DM stipulates that a qualified name can be mapped into an IRI by concatenating the IRI associated with the prefix and the local part.

A qualified name's prefix is OPTIONAL. If a prefix occurs in a qualified name, it refers to a namespace declared in a namespace declaration. In the absence of prefix, the qualified name refers to the default namespace.

6. PROV-DM Extensibility Points

The PROV data model provides extensibility points that allow designers to specialize it for specific applications or domains. We summarize these extensibility points here.

The PROV namespace declares a set of reserved attributes catering for extensibility: `prov:type`, `prov:role`, `prov:location`.

- Sub-types and sub-relations can be expressed by means of the reserved attribute `prov:type`.

Example 56◊

In the following example, `e2` is a translation of `e1`, expressed as a sub-type of derivation.

```
wasDerivedFrom(e2,e1, [prov:type='ex:Translation' ])
```

Example 57◊

In the following example, `e` is described as a Car, a type of entity.

```
entity(e, [prov:type='ex:Car' ])
```

- Application and domain specific roles can be expressed by means of the reserved attribute `prov:role`.

Example 58◊

In the following example, two computers `ex:laptop4` and `ex:desktop9` are used in different roles in a work activity.

```
activity(ex:work)
entity(ex:laptop4)
entity(ex:desktop9)
used(ex:work, ex:laptop4,  [prov:role="day-to-day machine"])
used(ex:work, ex:desktop9, [prov:role="backup machine"])
```

- Attribute-value lists occur in all types and most relations of the data model. Applications designers are free to introduce further application-specific attributes. Attributes for a given application can be distinguished by qualifying them with a prefix denoting a namespace declared in a namespace declaration. New namespaces and associated prefixes can be declared, allowing attributes and names to be qualified.

The PROV data model is designed to be application and technology independent, but implementers are welcome and encouraged to specialize PROV-DM to specific domains and applications. To ensure interoperability, specializations of the PROV data model that exploit the extensibility points summarized in this section must preserve the semantics specified in this document and in [PROV-CONSTRAINTS].

7. Creating Valid Provenance

This specification defines PROV-DM, a data model that allows descriptions of the people, institutions, entities, and activities, involved in producing, influencing, or delivering a piece of data or a thing to be expressed. However, with this data model, it is also possible to compose descriptions that would not make sense: for instance, one could express that an entity was used before it was generated, or that the activity that generated an entity started after the entity generation. A set of constraints have been defined for PROV and can be found in a companion specification [PROV-CONSTRAINTS]. They SHOULD be used by developers to compose provenance descriptions that are valid, and by implementers of reasoning engines aiming to check whether provenance descriptions have problems.

The example of section 3 contains identifiers such as `tr:WD-prov-dm-20111215`, which denotes a specific version of a technical report. On the other hand, an IRI such as http://www.w3.org/TR/prov-dm/ denotes the latest version of a document. One needs to ensure that provenance descriptions for the latter resource remain valid as the resource state changes.

To this end, PROV allows asserters to describe "*partial states*" of entities by means of attributes and associated values. Some further constraints apply to the use of these attributes, since the values associated with them are expected to remain unchanged for some period of time. The constraints associated to attributes allow provenance descriptions to be refined, they can also be found in the companion specification [PROV-CONSTRAINTS].

A. Cross-References to PROV-O and PROV-N

PROV-DM is a conceptual data model which can be serialized in various ways. The following table contains the PROV-O classes and properties, as described in [PROV-O], and PROV-N productions, as described in [PROV-N] that correspond to PROV-DM concepts.

Table 10◊: Cross-References to PROV-O and PROV-N

PROV-DM	PROV-O	PROV-N	Component
Entity	Entity	`entityExpression`	
Activity	Activity	`activityExpression`	

Generation	wasGeneratedBy, Generation	`generationExpression`	Component 1: Entities/Activities
Usage	used, Usage	`usageExpression`	
Communication	wasInformedBy, Communication	`communicationExpression`	
Start	wasStartedBy, Start	`startExpression`	
End	wasEndedBy, End	`endExpression`	
Invalidation	wasInvalidatedBy, Invalidation	`invalidationExpression`	
Derivation	wasDerivedFrom, Derivation	`derivationExpression`	Component 2: Derivations
Revision	wasRevisionOf, Revision	type `Revision`	
Quotation	wasQuotedFrom, Quotation	type `Quotation`	
Primary Source	hadPrimarySource, PrimarySource	type `PrimarySource`	
Agent	Agent	`agentExpression`	Component 3: Agents, Responsibility, Influence
Attribution	wasAttributedTo, Attribution	`attributionExpression`	
Association	wasAssociatedWith, Association	`associationExpression`	
Delegation	actedOnBehalfOf, Delegation	`delegationExpression`	
Plan	Plan	type `Plan`	
Person	Person	type `Person`	
Organization	Organization	type `Organization`	
SoftwareAgent	SoftwareAgent	type `SoftwareAgent`	
Influence	wasInfluencedBy, Influence	`influenceExpression`	
Bundle constructor	bundle description	`bundle`	Component 4: Bundles
Bundle type	Bundle	type `Bundle`	
Alternate	alternateOf	`alternateExpression`	Component 5: Alternate
Specialization	specializationOf	`specializationExpression`	
Collection	Collection	type `Collection`	Component 6: Collections
EmptyCollection	EmptyCollection	type `EmptyCollection`	
Membership	hadMember	`membershipExpression`	

B. Change Log

B.1 Changes since Proposed Recommendation

- Changed the status of this document section.
- Changed all URLs to PROV documents.
- ISSUE-653: Fixed typo in example to ensure compatibility with prov-n grammar.
- Fixed capitalization in definition of software agent, organization, and person.
- Fixed some typos and incorrect link.

B.2 Changes since Candidate Recommendation

- Checked all internal fragments resolved.
- Changed the status of this document section: added new documents to the PROV Family of Document, and removed the how to read section, referring instead to PROV-OVERVIEW.
- Changed all URLs to PROV documents.
- Fixed links to internal anchors, following change in respec.js
- Added anchors for prov:Bundle, prov:Collection, prov:Emptycollection, prov:Plan, prov:Person, prov:SoftwareAgent, prov:Organization, to facilitate systematic cross-referencing.
- Likewise, added anchor for Bundle Type.
- Table 9: fixed section number where plan is defined.
- Added html link to provenance.

B.3 Changes since Last Call

Please see the Responses to Public Comments on the Last Call Working Draft for more details about the justification of these changes.

- ISSUE-506: Updated role from author to contributor, in line with text.
- ISSUE-492: Fixed typos in Example 29.
- ISSUE-508: Clarified the bold names and parameters in text preceding Table 5.
- ISSUE-501: Put the example about driving a car to Boston in a box.
- ISSUE-450, ISSUE-514: added table with secondary objects in relations.
- ISSUE-512: simplied type of activity a2 to "fine paying"

- ISSUE-509: modified the introductory text to UML figures, so that they refer to relation names (e.g. WasStartedBy) as visualized in figures
- ISSUE-515: fixed typo
- ISSUE-531: added sentence on the use of prov:location attribute.
- ISSUE-519, ISSUE-523, ISSUE-524, ISSUE-529: changed UML diagram of figure 8 by removing explicit inheritance from influence for usage, start, end, generation, invalidation, communication, derivation, attribution, association, and delegation. Instead, introduced correspondance table 7. Furthermore, in response to these issues, it was made clear that PROV defines no attribute specific to subtypes such as SoftwareAgent, ..., Plan, Revision, Bundle, Collection.
- ISSUE-495: made explicit which section, figure, table was informative or normative.
- ISSUE-521: now states that "an agent relied on a plan" instead of "an agent adopted a plan".
- ISSUE-499: Made explicit that generation/usage/invalidation/start/end are implicit.
- ISSUE-449: Clarified definition of prov:value attribute and added an example.
- ISSUE-495: added paragraph about 'relations opening up'. Clarified the role of '-' in example. Fixed dates in biblio. Fixed space issue in prov-n examples
- ISSUE-516: Stating that there moust be some underpinning activity or activities for a derivation, instead of just activities.
- ISSUE-525: Made it explicit that Membership, Alternate, Specialization are not Influence
- Copied the sentence " An agent may be a particular type of entity or activity. This means that the model can be used to express provenance of the agents themselves. " from the informative section into the normative section.
- ISSUE-504: Updated definition of collection.
- ISSUE-503: Rephrased the introduction of expanded association in section 2.2.1.2 Expanded Relations.
- ISSUE-514: added links to the attributes listed in the secondary element table. Also removed PrimarySource, Quotation, Revision.
- ISSUE-502: Added sentence in section 2.1.2 explaining that the focus of derivation is on connecting a generated entity to a used entity.
- ISSUE-526: Added sentence clarifying sentence in section 5.5.2.
- ISSUE-462: Added clarification regarding entity attributes (with respect to fixed aspects) and role of identifier with respect to equality.
- ISSUE-518: Added clarifying sentence of primary source.
- ISSUE-552: Clarifying phrasing around a quotation/revision/primary-source relation is a particular case of a derivation relation ...; updated definitions for start and end.
- Rephrased original entity to preceding entity.
- Moved feature at risk, Mention, to note document (prov-mention).

C. Acknowledgements

This document has been produced by the Provenance Working Group, and its contents reflect extensive discussion within the Working Group as a whole. The editors extend special thanks to Sandro Hawke (W3C/MIT) and Ivan Herman (W3C/ERCIM), W3C contacts for the Provenance Working Group.

The editors acknowledge valuable contributions from the following: Tom Baker, David Booth, Robert Freimuth, Satrajit Ghosh, Ralph Hodgson, Renato Iannella, Jacek Kopecky, James Leigh, Jacco van Ossenbruggen, Alan Ruttenberg, Reza Samavi, and Antoine Zimmermann.

Members of the Provenance Working Group at the time of publication of this document were: Ilkay Altintas (Invited expert), Reza B'Far (Oracle Corporation), Khalid Belhajjame (University of Manchester), James Cheney (University of Edinburgh, School of Informatics), Sam Coppens (iMinds - Ghent University), David Corsar (University of Aberdeen, Computing Science), Stephen Cresswell (The National Archives), Tom De Nies (iMinds - Ghent University), Helena Deus (DERI Galway at the National University of Ireland, Galway, Ireland), Simon Dobson (Invited expert), Martin Doerr (Foundation for Research and Technology - Hellas(FORTH)), Kai Eckert (Invited expert), Jean-Pierre EVAIN (European Broadcasting Union, EBU-UER), James Frew (Invited expert), Irini Fundulaki (Foundation for Research and Technology - Hellas(FORTH)), Daniel Garijo (Universidad Politécnica de Madrid), Yolanda Gil (Invited expert), Ryan Golden (Oracle Corporation), Paul Groth (Vrije Universiteit), Olaf Hartig (Invited expert), David Hau (National Cancer Institute, NCI), Sandro Hawke (W3C/MIT), Jörn Hees (German Research Center for Artificial Intelligence (DFKI) Gmbh), Ivan Herman, (W3C/ERCIM), Ralph Hodgson (TopQuadrant), Hook Hua (Invited expert), Trung Dong Huynh (University of Southampton), Graham Klyne (University of Oxford), Michael Lang (Revelytix, Inc.), Timothy Lebo (Rensselaer Polytechnic Institute), James McCusker (Rensselaer Polytechnic Institute), Deborah McGuinness (Rensselaer Polytechnic Institute), Simon Miles (Invited expert), Paolo Missier (School of Computing Science, Newcastle university), Luc Moreau (University of Southampton), James Myers (Rensselaer Polytechnic Institute), Vinh Nguyen (Wright State University), Edoardo Pignotti (University of Aberdeen, Computing Science), Paulo da Silva Pinheiro (Rensselaer Polytechnic Institute), Carl Reed (Open Geospatial Consortium), Adam Retter (Invited Expert), Christine Runnegar (Invited expert), Satya Sahoo (Invited expert), David Schaengold (Revelytix, Inc.), Daniel Schutzer (FSTC, Financial Services Technology Consortium), Yogesh Simmhan (Invited expert), Stian Soiland-Reyes (University of Manchester), Eric Stephan (Pacific Northwest National Laboratory), Linda Stewart (The National Archives), Ed Summers (Library of Congress), Maria Theodoridou (Foundation for Research and Technology - Hellas(FORTH)), Ted Thibodeau (OpenLink Software Inc.), Curt Tilmes (National Aeronautics and Space Administration), Craig Trim (IBM Corporation), Stephan Zednik (Rensselaer Polytechnic Institute), Jun Zhao (University of Oxford), Yuting Zhao (University of Aberdeen, Computing Science).

D. References

D.1 Normative references

[PROV-CONSTRAINTS]
James Cheney; Paolo Missier; Luc Moreau; eds. *Constraints of the PROV Data Model*. 30 April 2013, W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-constraints-20130430/

[PROV-N]
Luc Moreau; Paolo Missier; eds. *PROV-N: The Provenance Notation*. 30 April 2013, W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-n-20130430/

[PROV-O]
Timothy Lebo; Satya Sahoo; Deborah McGuinness; eds. *PROV-O: The PROV Ontology*. 30 April 2013, W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-o-20130430/

[RDF-CONCEPTS]
Graham Klyne; Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. 10 February 2004. W3C Recommendation. URL: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels.* March 1997. Internet RFC 2119. URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC3987]
M. Dürst; M. Suignard. *Internationalized Resource Identifiers (IRIs) (RFC 3987)*. January 2005. RFC. URL: http://www.ietf.org/rfc/rfc3987.txt

[XMLSCHEMA11-2]
Henry S. Thompson et al. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. 5 April 2012. W3C Recommendation. URL: http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/

D.2 Informative references

[Logic]
W. E. Johnson. *Logic: Part III*.1924. URL: http://www.ditext.com/johnson/intro-3.html

[Mappings]
Satya Sahoo; Paul Groth; Olaf Hartig; Simon Miles; Sam Coppens; James Myers; Yolanda Gil; Luc Moreau; Jun Zhao; Michael Panzer; Daniel Garijo *Provenance Vocabulary Mappings*. August 2010 URL: http://www.w3.org/2005/Incubator/prov/wiki/Provenance_Vocabulary_Mappings

[PROV-AQ]
Graham Klyne; Paul Groth; eds. *Provenance Access and Query*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-aq-20130430/

[PROV-DC]
Daniel Garijo; Kai Eckert; eds. *Dublin Core to PROV Mapping*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-dc-20130430/

[PROV-DICTIONARY]
Tom De Nies; Sam Coppens; eds. *PROV Dictionary: Modeling Provenance for Dictionary Data Structures*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-dictionary-20130430/

[PROV-LAYOUT]
W3C PROV Working Group. *PROV Graph Layout Conventions*. 2012. URL: http://www.w3.org/2011/prov/wiki/Diagrams

[PROV-LINKS]
Luc Moreau; Timothy Lebo; eds. *Linking Across Provenance Bundles*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-links-20130430/

[PROV-OVERVIEW]
Paul Groth; Luc Moreau; eds. *PROV-OVERVIEW: An Overview of the PROV Family of Documents*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/

[PROV-PRIMER]
Yolanda Gil; Simon Miles; eds. *PROV Model Primer*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-primer-20130430/

[PROV-SEM]
James Cheney; ed. *Semantics of the PROV Data Model*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-sem-20130430.

[PROV-XML]
Hook Hua; Curt Tilmes; Stephan Zednik; eds. *PROV-XML: The PROV XML Schema*. 30 April 2013, W3C Note. URL: http://www.w3.org/TR/2013/NOTE-prov-xml-20130430/

[RDF-CONCEPTS11]
Richard Cyganiak; David Wood; eds. *RDF 1.1 Concepts and Abstract Syntax*. Working Draft. URL: http://www.w3.org/TR/rdf11-concepts/

[UML]
Object Management Group *Unified Modeling Language: Superstructure*. version 2.0, 2005 URL: http://www.omg.org/spec/UML/2.0/Superstructure/PDF/