# Provenance

## Introduction

Results of scientific experiments cannot be understood without the knowledge of the meaning of data and circumstances occurred during their creation. This type of knowledge includes data provenance (DAVIDSON; FREIRE, 2008; FREIRE *et al.*, 2008). Provenance is well understood in the context of art or digital libraries where historical documentation refers to an object’s life cycle (PREMIS WORKING GROUP, 2005). Recently, data provenance in scientific experimentation has become such an important topic that workshops and conferences on the subject were specifically created (SIMMHAN; PLALE; GANNON, 2005).

The *International Provenance and Annotation Workshop* (IPAW) (MOREAU *et al.*, 2002) was one of the first data provenance workshops to be created. In each edition, the scientific community listed challenges of data provenance to be solved and received many scientists work with possible solutions. During IPAW’06, participants were interested in questions about provenance for the usage in digital data, involving topics related to documentation, data annotation and data derivations (BOSE; FOSTER; MOREAU, 2006). As a result, the first model of digital provenance, the *Open Provenance Model* (OPM) (MOREAU *et al.*, 2007), was created. The OPM has been designed to address the issues raised during the *Provenance Challenge* (“Provenance Challenge WIKI”, 2010).

Later, another provenance model, PROV (GIL; MILES, 2010), was developed by the provenance incubator group W3C (GIL *et al.*, 2009). According to the group, provenance of digital objects represents the object’s origins and PROV is a proposed specification to represent these provenance records. These records contain descriptions of the entities and activities involved in producing and delivering or otherwise influencing a given object. The usage of provenance, regardless of the model, provides a critical foundation for assessing the authenticity of data, enabling reliability and reproducibility and is crucial component of workflow systems (GIL *et al.*, 2007; GROTH; MOREAU, 2010).

When PROV was proposed, the OPM model was already being used in several approaches. However, the fact that PROV is supported by the W3C makes the possibility of becoming the default provenance model, making the migration from OPM to PROV a possibility in the near future. With this, the aim of this chapter is to present a study of the digital provenance models, as well as comparing those models, pointing out their similarities and differences.

As such, this chapter is organized as follow: Section 3.2 and 3.3 describes the *Open Provenance Model* and PRO, respectively. Section 3.4 compares both digital provenance models and lastly, section 3.5 presents the final considerations of this chapter.

## Open Provenance Model

The *Open Provenance Model* emerged as a result from the *Provenance Challenges* proposed in the context of IPAW. The *Provenance Challenges* came in four editions, one for each year from 2006 to 2010 and OPM resulted from the first two challenges and was used on the third challenge:

*1st Challenge*: Aimed to provide a forum for the community to understand the capabilities of different provenance systems and express their provenance representations.

*2nd Challenge*: Aimed to establish interoperability between systems through exchange of provenance information.

*3rd Challenge*: Evaluate the OPM practically, from an inter-operability view-point.

The *Open Provenance Model* is a provenance model designed to meet the following requirements (MOREAU *et al.*, 2007):

* To allow provenance information to be exchanged between systems, by means of compatibility layer based on a shared provenance model.
* To allow developers to build and share tools that operates on such a provenance model.
* To define provenance in a precise, technology-agnostic manner.
* To support a digital representation of provenance for any “thing”, whether produced by computer systems or not.
* To allow multiple levels of descriptions to coexist.
* To define a core set of rules that identify the valid inferences that can be made on provenance representation.

In *Open Provenance Model*, it is assumed that provenance of objects is represented by an annotated causality graph, which is a directed acyclic graph enriched with annotations capturing further information pertaining to execution. According to MOREAU *et al.* (2007), a provenance graph is a record of a past or current execution, and not a description of something that could happen in the future.

### Types and Relations

The causality graph used by the *Open Provenance Model* is composed of vertices, which can represent *Artifacts*, *Processes* and *Agents*, and edges that represent causal relationships between vertices. Below are the definitions for all three node types:

***Artifacts*** are an immutable piece of state that can represent a physical object or a digital representation in a computer system.

***Processes*** are actions or a sequence of actions performed or caused by artifacts and results in new artifacts.

***Agents*** are contextual entities acting as a catalyst of a process that can enable, facilitate, control or affect its execution.

The edges of the graph belong to one of the categories described in , representing a causal dependency between its source, denoting the effect, and its destination that denotes the cause. Below are some important definitions in the Open Provenance Model according to MOREAU *et al.* (2007).

# 

Figure 1: Edges in OPM. Source: (MOREAU *et al.*, 2007).

**Causal Relationship**: Represented by an arc and denotes the presence of a causal dependency between the source (effect) and the destination (cause).

**Artifact Used by a Process**: A [*used*] edge from *process* to an *artifact* is a causal relationship intended to indicate that the *process* required the availability of the *artifact* to be able to complete its execution. When several *artifacts* are connected to a same *process* by multiple [*used*] edges, all of them were required for the *process* to complete.

**Artifacts Generated by Processes**: A [*was generated by*] edge from an *artifact* to a *process* is a causal relationship intended to mean that the *process* was required to initiate its execution in order to generate the *artifact*. When several *artifacts* are connected to the same *process* by multiple [*was generated by*] edges, the *process* must begin for all of them to be generated.

**Process Triggered by Process**: An edge [*was triggered by*] from a *process* P2 to a *process* P1 is a causal dependency that indicates that the start of *process* P1 was required for P2 to be able to complete.

**Artifact Derived from Artifact**: An edge [*was derived from*] from *artifact* A2 to *artifact* A1 is a causal relationship that indicates that *artifact* A1 should have been generated for A2 to be generated. The piece of state associated with A2 is dependent on the presence of A1 or on the piece of state associated with A1.

**Process Controlled by Agent**: An edge [*was controlled by*] from a *process* P to an *agent* Ag is a causal dependency that indicates that *agent* Ag controlled the start and end of *process* P.

**Role**: Designates an *artifact* or *agent's* function in a *process*.

In , the edge [*used*] say that a *process* used an *artifact*, while the [*was generated by*] edge an *artifact* was generated by a *process*. The letter "R" represents the roles under which these *artifacts* were used since a *process* may have used several *artifacts*. Likewise, many *artifacts* may have been generated by a *process*, and each would have a specific role. Roles are only meaningful in the context of the *process* where they are defined, and they are not defined by the OPM itself, but by the application domains. Roles are used on OPM just to distinguish the involvement of *artifacts* in *processes*.

The edge [*was controlled by*] means the *process* was caused by an *agent*, essentially acting as a catalyst or controller. Since a *process* may have been controlled by several *agents*, their roles are also identified as controllers. This type of dependency represents a control relationship and not a data derivation. The edge [*derived from*] assert that *artifact* A2 was derived from another *artifact* A1, giving an oriented dataflow view of the provenance. In contrast to the edge [*was derived from*], an edge [*was triggered by*] allows a *process* to have an oriented view of past executions.

### Time Information

Moreover, the Open Provenance Model allows causality graphs to be used with time information. In this model, time is not used for deriving causality, but to validate causality claims, since if the same time clock is used to measure the time for both the effect and cause, then the time of an effect should be greater than the time of its cause.

In addition, time may be associated to *instantaneous occurrences* in a *process*. There are four types of this occurrences, being denoted as *creation* and *use* for *artifacts* and *starting* and *ending* for *processes*. Given that time may be observed by someone, its accuracy is limited by the clock and the notion of time. This way, the model allows for an interval of accuracy to support the granularity used to represent time. With this, it is possible to state that an *artifact* was used no earlier than time t1 and no later than time t2, as an example. This rationale is analogous for *processes*.

indicates how time information can be expressed in the model. For [*used*] and [*was generated by*] edges, one timestamp can be used to express when the event happened. For [*was controlled by*] edge two timestamps marks when the process started and terminated. For [*was derived from*] and [*was triggered by*] edges, one timestamp to indicate when the *artifact* was used. Despite using timestamp, the time of occurrence itself is not enough to imply causality. The fact that *process* P1 happened before P2 is not enough information to infer that P1 caused P2 to happen.

### Completion Rules

Finally, the Open Provenance Model has defined the notion of a graph based on a set of syntactic rules and topological constraints. The provenance graph captures causal dependencies that can be summarized by means of transitive closure. Because of this, a set of completion rules and inferences can be used in the graph.

For completion rules, there is the *artifact elimination*, also known as forward transformation. shows such transformation. The edge [*was triggered by*] can be obtained from the existence of [*used*] and [*was generated by*] edges. Also in the same figure, there is another completion rule, called *artifact introduction*, which establishes that the [*was triggered by*] edge is hiding the existence of an *artifact* used by P2 and generated by P1. The completion rules allow the establishment of the existence of some *artifacts* but it does not make explicit their identities. This is the consequence of using [*was triggered by*], which is a composition of [*used*] and [*was generated by*]. On the other hand, presents a completion rule regarding *process introduction*. The edge [*was derived from*] hide the presence of an intermediary *process*. However, the converse rule does not work without some internal knowledge of P, which is fundamental to ascertain if there is an actual dependency between A1 and A2.



Figure 2: Artifact introduction and elimination. Source: (MOREAU *et al.*, 2007).

When users want to find out the causes of an *artifact* or a *process*, their interest is in indirect causes that involve multiple transitions. For this purpose, a set of new relationships was created:

**Multi-step "wasDerivedFrom"**: An *artifact a1* was derived from *A2* (possibly using multiple steps), written as *a1🡪\* a2*, if *a1* was derived from *a2* or from an *artifact* that was itself derived from *a2* (possibly using multiple steps). In other words, it is the transitive closure of the edge [*was derived from*]. It expresses that *artifact* *a2* had an influence on *artifact a1.*



Figure 3: Process introduction. Source: (MOREAU *et al.*, 2007).

**Secondary Multi-Step Edges**:

**Process *p* used artifact *a* (possibly using multiple steps)**:written as *p 🡪\* a*, if *p* used an *artifact* *a* or an *artifact* that derived *a* (possibly using multiple steps).

**Artifact *a* was generated by process *p* (possibly using multiple steps)**:written as *a* 🡪\* *p*, ifa or an *artifact* that derived *a* (possibly using multiple steps) that was generated by *p.*

**Process *p1* was triggered by process *p2* (possibly using multiple steps)**:written as *p1* 🡪\* *p2,* if *p1* used an *artifact* that was generated or was derived from an *artifact* (possibly using multiple steps) that was itself generated by *p2.*

Multi-step edges can be inferred from single step edges by eliminating *artifacts* that occur in chains of dependencies. Analyzing , it is possible to infer that *process* *p2* was triggered by *p1*, omitting the fact that *p2* used *a3*, which was derived from *a2* that in turn was derived from *a1*, which was generated by *p1*. Other inferences are also illustrated in .

Lastly, the *Open Provenance Model* has a modular design as illustrated by Figure 5. However, specifications for all layers in the design have not been produced yet. At the bottom layer is located the abstract model (MOREAU *et al.*, 2007). On the left side, a serialization to *xml*, defined by OPMX (The Open Provenance Model XML Schema) (MOREAU; GROTH; *et al.*, 2010), and a mapping to RDF with OPMV (The Open Provenance Model Vocabulary) (ZHAO, 2010) and OPMO (The Open Provenance Model OWL Ontology) (MOREAU; DING; *et al.*, 2010). Those are the only specifications produced, along with the *Open Provenance Model Java Library* (MOREAU, 2010b), a JAXB-generated Java Library used by *OPM Toolbox* (MOREAU, 2010a) for creating a Java representation of OPM graphs and serializing them to or from a *xml*  file. With the development of PROV, these other OPM specifications were left unfinished.



Figure 4: Inference. Source: (MOREAU *et al.*, 2007).

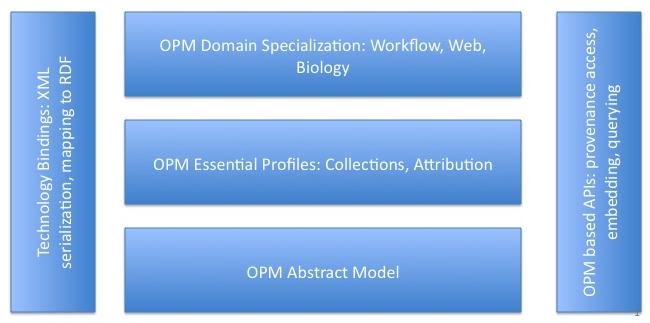


Figure 5: OPM’s Layered Architecture. Source: (MOREAU *et al.*, 2007)

## PROV

PROV is a family of specifications proposed by W3C group to express provenance of digital objects, containing descriptions of the entities and activities involved in producing and delivering an object. In their view, provenance is information about entities, activities, and people involved in producing a piece of data which can be used to form assessments about its quality, reliability or trustworthiness. The goal of this provenance model is to enable the wide publication and interchange of provenance on the *web* and other information systems. The PROV model enables the representation and interchange of provenance information using widely known and available formats such as *RDF* and *xml* (GROTH; MOREAU, 2010).

The discussion group that gave developed PROV was officially launched concurrently to the forth *Provenance Challenge* (“Provenance Challenge WIKI”, 2010). The specifications that make up the provenance model PROV were divided into several documents that details different aspects of it: PROV *Overview (PROV-OVERVIEW)* (GROTH; MOREAU, 2010), PROV *Primer (PROV-PRIMER)* (GIL; MILES, 2010), Prov *Ontology (PROV-O)* (LEBO; SAHOO; MCGUINESS, 2010), PROV *Data Model (PROV-DM)* (MOREAU; MISSIER, 2010a), PROV *Constraints (PROV-CONSTRAINTS)* (NIES *et al.*, 2010), PROV *Notation (PROV-N)* (MOREAU; MISSIER, 2010b), PROV XML (PROV-XML) (HUA *et al.*, 2010), PROV *Dublin Core Mapping (PROV-DC)* (GARIJO *et al.*, 2010), PROV *Links* (MOREAU; LEBO, 2010), PROV *Access and Query (PROV-AQ)* (WEITZNER *et al.*, 2008), PROV *Dictionary (PROV-DICTIONARY)* (MISSIER *et al.*, 2010), PROV *Semantics (PROV-SEM)* (CHENEY, 2010) and PROV *Implementations* (GROTH *et al.*, 2012) .

Figure 6 illustrates the organization of PROV. At its core, is a conceptual data model which defines a common vocabulary used to describe provenance. To help developers and users, a set of constraints are defined to create provenance validators[[1]](#footnote-1). Lastly, to support the interchange of provenance, other definitions are provided for protocols to locate and access provenance, connect sets of provenance and define how to interoperate.

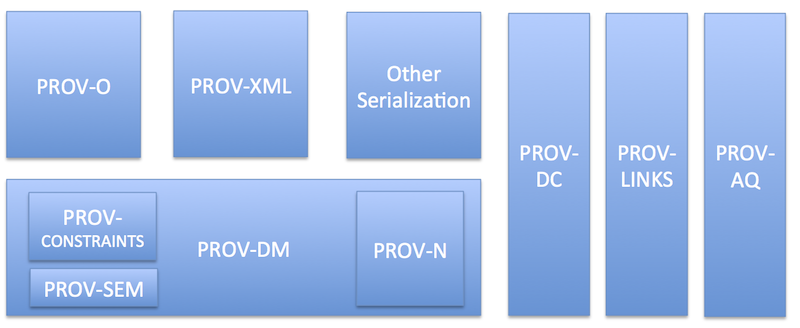


Figure 6: PROV organization. Source: (GROTH; MOREAU, 2010)

Provenance can be used for many purposes, such as understanding how the data was collected in order to use it meaningfully, determining the object’s ownership and rights, making judgments about the information to determine whether to trust it. It can also verify the process and steps used to obtain the result complies with given requirements. Lastly, to reproduce how something was generated. As a specification for provenance, the PROV model accommodates all those uses of provenance. However, different people may have different perspectives on provenance. Because of this, there are three different types of information that might be captured in provenance records:

***Agent-centered provenance***: describes which entities were involved in generating or manipulating the information in question.

***Object-centered provenance***: traces the origins of portions of a document to other documents.

***Process-centered provenance***: captures the actions and steps taken to generate the information in question.

### Types and Notations

PROV also uses a graph, similar to the provenance graph from OPM, to represent provenance information. This graph is also characterized by having edges representing relationships between vertices and three types of vertices: *Entities*, *Activities*, and *Agents*.

***Entities***: physical, digital, conceptual, or other kinds of things. Examples are web pages, charts and spellcheckers. They may also be described as having different attributes and be described from different perspectives.

***Activities***: how *entities* came into existence and how their attributes changed to become new *entities*, often making use of previously existing *entities*. *Activities* are dynamic aspects of the world, such as actions and processes.

***Agents***: person, a piece of software, an inanimate object, an organization, or other *entities* that may be ascribed responsibility. An *agent* takes a role in an *activity* such that the *agent* can be assigned some degree of responsibility for the *activity* taking place.

When an *agent* has some responsibility for an *activity*, that *agent* was associated with the *activity*. Several *agents* may be associated with an *activity* and vice-versa. An *agent* may also be acting on behalf of other *agents*. Such types of relations are represented by edges in the provenance graph. Figure 7 illustrates those vertices types and their respective shapes in the graph along with some possible relations between them. These relations, as well as other possible relations, are defined below according to PROV-DM (MOREAU; MISSIER, 2010a):

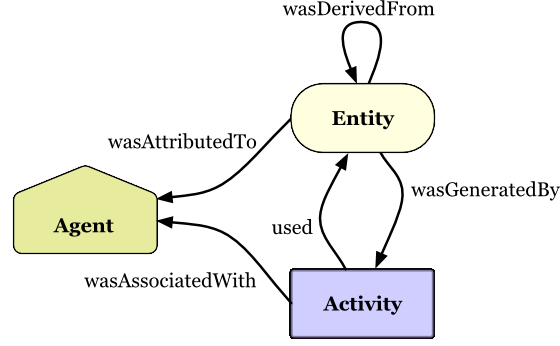


Figure 7: PROV Entities and relations. Source: (GIL; MILES, 2010)

**Usage**: A [*used*] edge from *activity* to an *entity* indicates that is the beginning of utilizing an *entity* by an *activity*. Before *used*, the *activity* had not begun to utilize the *entity* and could not have been affected by it.

**Start**: A [*wasStartedBy*] edge indicates 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 the *activity* follows the *activity’s* start. A start may refer to a trigger *entity* that set off the *activity*, or to another *activity*, known as **starter,** that generated the trigger.

**End**: A [*wasEndedBy*] edge indicates 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 the *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.

**Generation**: A [*wasGeneratedBy*] edge from an *entity* to an *activity* indicates that the *entity* was generated by the *activity*. The *entity* did not exist before generation and becomes available for usage after this generation.

**Invalidation**: A [*wasInvalidadedBy*] edge is the start of the destruction, cessation, or expiry of an existing *entity* by an *activity*. The *entity* is no longer available for use after invalidation. Any generation or usage of an *entity* precedes its invalidation.

**Communication**: A [*wasInformedBy*] edge is the exchange of some unspecified *entity* between two *activities*, one *activity* using some *entity* generated by the other *activity*.

**Derivation**: A [*wasDerivedFrom*] edge is the transformation of an *entity* into another, an update of the *entity* resulting in a new one, or the construction of a new *entity* based on a pre-existing *entity*.

**Attribution**: The [*wasAttributedTo*] edge from an *entity* to an *agent* is the ascribing of the *entity* to the *agent*.

**Association**: A [*wasAssociatedWith*] edge from an *activity* to an *agent* is an assignment of responsibility to the *agent* for the *activity*, indicating that the *agent* had a role in the *activity*.

**Delegation**: The [*actedOnBehalfOf*] edge from *agent* to another indicates the assignment of authority and responsibility to the *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.

**Revision**: A [*wasRevisionOf*] edge indicates a derivation for which the resulting *entity* is a revised version of the original *entity*.

**Quotation**: A [*wasQuotedFrom*] edge indicates the repeat of an *entity*, such as text or image, by someone who may or may not be its original author.

**Influence**: A [*wasInfluencedBy*] edge indicates that the *entity, activity* or *agent* had an effect on the character, development, or behavior of another by the means of *usage, start, end, generation, invalidation, communication, derivation, attribution, association,* or *delegation.*

### Further Notations

Besides the relations mentioned in the previous subsection, the PROV model has support for a few more: Specialization, Alternate, and the possibility of extending existing structures. These extended structures are defined by a variety of mechanisms: *subtyping*, *expanded relations*, *optional identification*, and *new relations*.

***Specialization***: A [*specializationOf*] edge from an *entity* to another indicates that the first *entity* 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. A specialization is not defined as an influence.

***Alternate***: A [*alternateOf*] edge from an *entity* to another indicates that both of them 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. Alternate is not defined as an influence. The alternate relationship is a necessary general relationship that only states that both alternate *entities* respectively fix some aspects of some common thing, and so there is some relevant connection between the provenances of the alternates.

***Subtyping***: can be applied to core types. For example, a *software* *agent* is special kind of *agent*. *Subtyping* can also be applied to core relations. For example, a *revision* is a special kind of *derivation*: *revision* is a *derivation* for which the resulting *entity* is a revised version of the original.

***Expanded Relations***: Binary relations can be expanded by applications and filled in with further application details. For example, in a *derivation* relationship, the application may decide to expand that relationship in order to describe how the *entity* was derived from another. Another example is with *agents* who may rely on *plans*, which are defined as a set of actions or steps necessary to achieve their goals in the context of an *activity*. A *plan* is defined by *subtying*: A *plan* is an *entity* that represents a set of actions or steps intended by one or more *agents* to achieve some goals. Figure 8 illustrates the usage of the *expanded relation* *plan* in the [*hadPlan*] edge connecting the edge [*wasAssociatedWith*], from an *agent* (edith) and an *activity* (correct), with an *entity* (instructions).

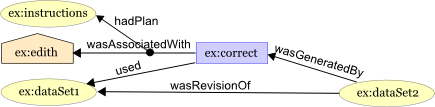


Figure 8: Using *Expanded Relations*. Source: (GIL; MILES, 2010)

***Optional Identification***: The PROV model also allows for an optional identifier to express an instance of an association between two or more elements. This option identifier can then be used to refer to an instance as part of other concepts.

***Further Relations***: The PROV model also supports further relations that are not *subtypes* or *expanded versions* of existing relations. For example, *specialization* and *alternate* can be considered new relations. Figure 9 illustrates the usage of further relations (*specialization* and *alternate*), as well as an optional identification (“Crime rises in cities”) from the *entity* “article”.

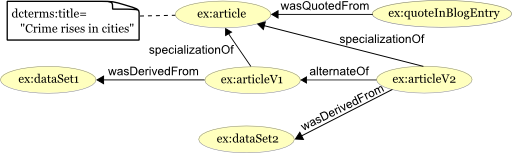


Figure 9: *Optional ID* and *Further Relations*. Source: (GIL; MILES, 2010)

The PROV data model also has a set of pre-defined attributes that can be used to provide further details. These attributes are optional and can be up to five different types: *label*, *location*, *role*, *type*, and *value*.

***Label***: provides a human-readable representation of an instance of types (*agents*, *entity*, and *activity*) or relationships.

***Location***: provides an identifiable place, for example: geographic, directory, row, column, address, landmark, coordinates, and so forth.

***Role***: provides the function of an *entity* or *agent* with respect to an *activity*.

***Type***: provides further typing information for any construct with an optional set of attribute-value pairs. Example: Bundle, collection, organization, person, *plan*, *softwareAgent*.

***Value***: provides a value that is a direct representation of an *entity*. A *value* is a constant such as string, number, time, qualified name, encoded binary data, and so forth. Table 1 describes which constructs are allowed the usage of attributes and if there is any restriction of its value.

Table : PROV optional attributes

|  |  |  |
| --- | --- | --- |
| Attribute | Allowed in | Value |
| Label | Any constructs | *Value* of type *String* |
| Location | *Entity*, *Activity*, *Agent*, *Usage*, *Generation*, *Invalidation*, *Start*, and *End* | *Value* |
| Role | *Usage*, *Generation*, *Invalidation*, *Association*, *Start* and *End* | *Value* |
| Type | Any constructs | *Value* |
| Value | *Entity* | *Value* |

### Time Information

The PROV model offers the ability to store information data from the time of origin due to the importance of temporal information in some scenarios. It is allowed to store date and time relating to *entities* or *activities*. For *entities*, is allowed to store information from its generation or usage. As for *activities*, it is allowed to store information from when it started and ended its execution.

This information can be stored in tickets in the *activity* or in the relationships, as illustrated by Figure 10, showing the *startedAtTime* and *endedAtTime* in the *activity* “correct”, and the *generatedAtTime* in the relationships *wasGeneratedBy*. These tickets can also be used to store other information details, as in the usage of *Optional Identification* as mentioned in the previous subsection.

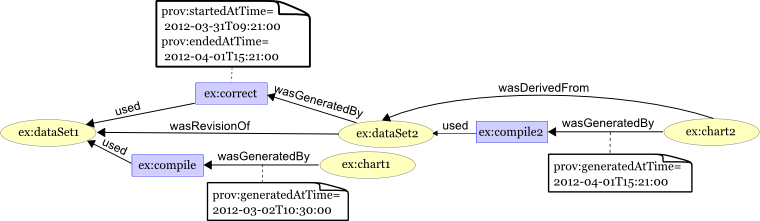


Figure 10: Time information. Source: (GIL; MILES, 2010)

### Inference

Like OPM, the PROV model also supports the usage of inferences on provenance data, preserving *equivalence* on *valid* PROV instances. A PROV instance is *valid* if its *normal* *form* exists and all of the validity constraints are true on the *normal* *form*. PROV defines the *normal form* of a PROV instance as the set of provenance statements resulting from applying all definitions, inferences, and uniqueness constraints. This can be obtained as follows (NIES *et al.*, 2010):

1. Apply all definitions to instance *I* by replacing each defined statement by its definition, yielding an instance *I1*.
2. Apply all inferences to *I1* by adding the conclusion of each inference whose hypotheses are satisfied and whose entire conclusion does not already hold, yielding an instance *I2*.
3. Apply all uniqueness constraints to *I2* by unifying terms or merging statements and applying the resulting substitution to the instance, yielding an instance *I3*. If some uniqueness constraint cannot be applied, then normalization fails.
4. If no definitions, inferences, or uniqueness constraints can be applied to instance *I3*, then *I3* is the normal form of *I*.
5. Otherwise, the normal form of *I* is the same as the normal form of *I3* (that is, proceed by normalizing *I3* at step 1).

So, in order to test the PROV instance validity, the following steps must be followed (NIES *et al.*, 2010):

1. Normalize the instance *I*, obtaining normal form *I'*. If normalization fails, then *I* is not valid.
2. Apply all event ordering constraints to *I'* to build a graph *G* whose vertices are event identifiers and edges are labeled by "precedes" and "strictly precedes" relationships among events induced by the constraints.
3. Determine whether there is a cycle in *G* that contains a "strictly precedes" edge. If so, then *I* is not valid.
4. Apply the *type constraints[[2]](#footnote-2)* to determine whether there are any violations of disjunction. If so, then *I* is not valid.
5. Check that none of the *impossibility constraints[[3]](#footnote-3)* are violated. If any are violated, then *I* is not valid. Otherwise, *I* is valid.

Finally, two *valid* PROV instances are *equivalent* if they have an isomorphic normal form, which means that after applying all possible inference rules, both instances produce the same set of PROV statements. Equivalence can also be checked by pairs of PROV instances that are not *valid*, according to the following rules (NIES *et al.*, 2010):

* If both are valid, then equivalence is defined above.
* If both are invalid, then equivalence can be implemented in any way provided it is reflexive, symmetric, and transitive.
* If one instance is valid and the other is invalid, then the two instances are not equivalent.

In *equivalence*, the order of provenance statements is irrelevant to the meaning of the instance. The order of attributes and values pair in the attribute lists is also irrelevant. Names can also be renamed without changing the meaning, so particular choices of names of existential variables are also irrelevant. Finally, *equivalence* is reflexive, symmetric, and transitive.

An *inference* in PROV is a rule that can be applied to PROV instances to add new statements, while a *definition* is a rule that can be applied to instances to replace defined statements with other statements. In other words, a *definition* states that a provenance statement is equivalent to some other statements, while an *inference* only states one direction of an implication. *Definitions* and *inferences* can also be viewed as logical formulas (NIES *et al.*, 2010).

## Comparison Between Models

In terms of key elements from both provenance models, it is possible to make a direct mapping between key concepts by associating *artifacts*, *process* and *agents* from OPM to *entities*, *activities* and *agents* in PROV, respectively. Both models present ways of marking the passage of time and execution, as well as providing rules for making inferences. However, PROV also provide explicit support for extending existing features, such as *subtyping*, expanding, and creating new relationships. Some relationships from both models are also compatible because, aside from having the same names, they also carry the same causal relationship between objects. These common relationships are: *used*, *wasGeneratedBy*, and *wasDerivedFrom*.

However, the relationship *wasControlledBy* from OPM doesn’t have one from PROV with the same name, but the relationship *wasAssociatedWith* from PROV has the same function, linking *activities* (*processes* in OPM) to *agents* in both models. The relationship *wasTriggedBy* is a relationship between two *processes* in OPM and despite PROV also having a relationship between two *activities* (equivalent to *processes* in OPM) among its set of relationships, the relationship *wasInformedBy* has a different purpose. This relationship aims to show that a particular *activity* reported something to the other, while *wasTriggedBy* from OPM indicates that a *process* has been initiated by another. However, there is a relationship in PROV (*wasStartedBy*) which equals to *wasTriggedBy* from OPM. Although the relationship *wasStartedBy* is more comprehensive as it can occur not only between two *activities*, but also between an *entity* and an *activity*.

Also, the PROV model has four relationships that were not found in OPM: the aforementioned *wasInformedBy*, *wasEndedBy*, *actedOnBehalfOf*, and *wasAttributedTo*. The relationships *actedOnBehalfOf* and *wasAttributedTo* are delegation and association of *agents* to *entities* and *activities*. These are extremely important because PROV aim to provide provenance information also centered on *agents*, something that does not occur naturally in OPM. Lastly, the relationship *wasEndedBy* aims to represent the *activity’s* finalization. Table 2 illustrates the comparison of the existing relationships from both provenance models.

From these relationships without direct equivalences, it is possible to observe differences between models. The OPM is a simpler, and apparently is aimed to control flows of execution taking particular indication of a *process* being started by another. Meanwhile, PROV appears to be more focused on issues of responsibility and historical data, having several relationships between *agents* and the other types (*entities* and *activities*), but also being more complete, having all relationships equivalent to the OPM. This may be due to the fact that the majority of OPM’s designers also participated in the creation of PROV.

Table : OPM x PROV

|  |  |
| --- | --- |
| OPM | PROV |
| *used* | ***used*** |
| *wasGeneratedBy* | ***wasGeneratedBy*** |
| *wasControlledBy* | ***wasAssociatedWith*** |
| *wasDerivedFrom* | ***wasDerivedFrom*** |
| *wasTriggeredBy* | ***wasStartedBy*** |
|  | ***wasEndedBy*** |
|  | ***wasRevisionOf*** |
|  | ***wasAttributtedTo*** |
|  | ***wasInformedBy*** |
|  | ***actedOnBehalfOf*** |

## Final Considerations

In this chapter was presented the concepts of provenance in order to gather historical information about objects for further analysis. It was also presented both the existing provenance models (OPM and PROV) that can be used for provenance of digital information. Later, a comparison between models was made, pointing out their similarities. It might be also possible to attribute the lack of documentation for OPM due to the fact that the same designers were involved in the creation of PROV, which occurred around the same year that OPM updates halted (at 2010). By analyzing both provenance models, there are three key points that led to the construction of a new approach of game flow analysis for games:

* Provenance of objects, which allows for a detailed study of an object’s life cycle.
* Provenance inferences, which allows making statements while at the same time hiding unimportant facts to reach conclusions about the object’s history.
* Provenance Graph, which allows for an analysis of the object’s interactions and influences from other *entities* throughout its life cycle.

With this, it is proposed a new approach to improve the player’s understanding of the game flow, providing insights on how the story progressed and what influenced in the outcomes. In order to improve understanding, it is provided the means to analyze the game flow by using provenance. This new approach, called *Provenance in Games*, is presented in Chapter 4.

# References

BOSE, Rajendra; FOSTER, Ian; MOREAU, Luc. Report on the International Provenance and Annotation Workshop: (IPAW’06) 3-5 May 2006, Chicago. *SIGMOD Rec.*, v. 35, n. 3, p. 51–53, set. 2006. Accessed: 2 abr. 2013.

CHENEY, James. *Semantics of the PROV Data Model*. Available: <http://www.w3.org/TR/2013/WD-prov-sem-20130312/>. Accessed: 26 mar. 2013.

DAVIDSON, Susan B.; FREIRE, Juliana. Provenance and scientific workflows: challenges and opportunities. SIGMOD ’08, 2008, New York, NY, USA. *Anais*... New York, NY, USA: ACM, 2008. p. 1345–1350. Available: <http://doi.acm.org/10.1145/1376616.1376772>. Accessed: 26 mar. 2013.

FREIRE, J. *et al.* Provenance for Computational Tasks: A Survey. *Computing in Science Engineering*, v. 10, n. 3, p. 11–21, 2008.

GARIJO, Daniel *et al.* *Dublin Core to PROV Mapping*. Available: <http://www.w3.org/TR/2013/WD-prov-dc-20130312/>. Accessed: 26 mar. 2013.

GIL, Yolanda *et al.* Examining the Challenges of Scientific Workflows. *Computer*, v. 40, n. 12, p. 24–32, dez. 2007. Accessed: 2 abr. 2013.

GIL, Yolanda *et al.* *W3C Provenance Incubator Group*. Available: <http://www.w3.org/2005/Incubator/prov/wiki/Main\_Page>. Accessed: 22 mar. 2013.

GIL, Yolanda; MILES, Simon. *PROV Model Primer*. Available: <http://www.w3.org/TR/prov-primer/>. Accessed: 21 mar. 2013.

GROTH, Paul *et al.* *ProvImplementations*. Available: <http://www.w3.org/2011/prov/wiki/ProvImplementations>. Accessed: 26 mar. 2013.

GROTH, Paul; MOREAU, Luc. *PROV-Overview*. Available: <http://www.w3.org/TR/prov-overview/>. Accessed: 26 mar. 2013.

HUA, Hook *et al.* *PROV-XML: The PROV XML Schema*. Available: <http://www.w3.org/TR/prov-xml/>. Accessed: 26 mar. 2013.

LEBO, Timothy; SAHOO, Satya; MCGUINESS, Deborah. *PROV-O: The PROV Ontology*. Available: <http://www.w3.org/TR/prov-o/>. Accessed: 21 mar. 2013.

MISSIER, Paolo *et al.* *PROV Dictionary*. Available: <http://www.w3.org/TR/2013/WD-prov-dictionary-20130312/>. Accessed: 26 mar. 2013.

MOREAU, Luc *et al.* *IPAW*. Available: <http://www.ipaw.info/>. Accessed: 2 abr. 2013.

MOREAU, Luc; DING, Li; *et al.* *Open Provenance Model (OPM) OWL Specification*. Available: <http://openprovenance.org/model/opmo>. Accessed: 2 abr. 2013.

MOREAU, Luc; GROTH, Paul; *et al.* *Open Provenance Model (OPM) XML Schema Specification*. Available: <http://openprovenance.org/model/opmx>. Accessed: 2 abr. 2013.

MOREAU, Luc. *OPM Toolbox*. Available: <http://openprovenance.org/toolbox.html>. Accessed: 2 abr. 2013a.

MOREAU, Luc. *OPM4J: The Open Provenance Model Java Library*. Available: <http://openprovenance.org/java/site/1\_1\_8/apidocs/org/openprovenance/model/package-summary.html>. Accessed: 2 abr. 2013b.

MOREAU, Luc *et al.* The Open Provenance Model core specification (v1.1). *In: Future Generation Computer Systems*, v. 27, n. 6, p. 743–756, 2007. Accessed: 10 jun. 2012.

MOREAU, Luc; LEBO, Timothy. *Linking Across Provenance Bundles*. Available: <http://www.w3.org/TR/2013/WD-prov-links-20130312/>. Accessed: 26 mar. 2013.

MOREAU, Luc; MISSIER, Paolo. *PROV-DM: The PROV Data Model*. Available: <http://www.w3.org/TR/prov-dm/>.

MOREAU, Luc; MISSIER, Paolo. *PROV-N: The Provenance Notation*. Available: <http://www.w3.org/TR/prov-n/>. Accessed: 21 mar. 2013b.

NIES, Tom De *et al.* *Constraints of the PROV Data Model*. Available: <http://www.w3.org/TR/prov-constraints/>. Accessed: 21 mar. 2013.

PREMIS WORKING GROUP. *Data Dictionary for Preservation Metadata*. Technical, no Final Report of the PREMIS Working Group. Preservation Metadata: Implementation Strategies (PREMIS), 2005. Available: <http://www.oclc.org/research/projects/ pmwg/premis-final.pdf>.

*Provenance Challenge WIKI*. Available: <http://twiki.ipaw.info/bin/view/Challenge/>. Accessed: 26 mar. 2013.

SIMMHAN, Yogesh L.; PLALE, Beth; GANNON, Dennis. A survey of data provenance in e-science. *SIGMOD Rec.*, v. 34, n. 3, p. 31–36, set. 2005. Accessed: 2 abr. 2013.

WEITZNER, Daniel J. *et al.* Information accountability. *Communications of the ACM*, v. 51, n. 6, p. 82–87, 1 jun. 2008. Accessed: 21 mar. 2013.

ZHAO, Jun. *Open Provenance Model Vocabulary Specification*. Available: <http://open-biomed.sourceforge.net/opmv/ns.html>. Accessed: 2 abr. 2013.

1. A validator is a [computer program](http://en.wikipedia.org/wiki/Computer_program) used to check the [validity](http://en.wikipedia.org/wiki/Validity) or syntactical correctness of a fragment of code or document. [↑](#footnote-ref-1)
2. *Type constraints* are defined at section 6.3 of PROV-CONSTRAINTS (NIES *et al.*, 2010) [↑](#footnote-ref-2)
3. *Impossibility constraints* are defined section 6.4 of PROV-CONSTRAINTS (NIES *et al.*, 2010) [↑](#footnote-ref-3)