# - Provenance Models

## 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). However, for digital provenance there are two types of provenance: retrospective and prospective (FREIRE *et al.*, 2008). The prospective provenance focuses on specifications and the necessary steps to achieve the generated data, while retrospective provenance focuses on the executed steps and external information used to derive the data.

Recently, data provenance in scientific experimentation has become an important topic in scientific research and, as consequence, 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 lists challenges of data provenance to be solved and receives scientific works 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* (MILES *et al.*, 2010), such as understanding provenance systems and how to express provenance information, and is based on retrospective provenance.

Later, another provenance model, PROV (GIL; MILES, 2010), was developed by the provenance incubator group at 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. PROV also is focused on retrospective provenance, similar to OPM. The usage of provenance, regardless of the model, provides a critical foundation for assessing the authenticity of data, enabling reliability and reproducibility, and is a 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 and the migration from OPM to PROV a possibility in the near future. Nevertheless, the aim of this chapter is to present a study of both digital provenance models, which uses retrospective provenance, as well as comparing those models, pointing out their similarities and differences.

As such, this chapter is organized as follow: Section 3.2 provides a guiding example to aid understanding of provenance relationships in a provenance graph. Section 3.3 and 3.4 describe the OPM and PROV, respectively. Section 3.5 compares both digital provenance models and, lastly, section 3.6 presents the final considerations of this chapter.

## Guiding Example

Cake is often a desired dessert for receiving guests or special occasions and is generally easy to make. There are countless varieties of cakes but they tend to follow the same basic steps: Mixing, baking, and decoration. The first step, mixing, is responsible of mixing the cake’s ingredients in order to make cake batter. Basic ingredients used in this stage are usually butter, flour, sugar, and eggs. After mixing the ingredients for a period of time, generally until the cake batter acquires a uniform color, the batter goes to a cake pan in order to undergo the next stage, baking.

The baking stage takes the cake to the oven to bake the batter. Generally the baking process takes around 30 to 45 minutes, depending on the oven. After the baking is complete, the cake must be left untouched to cool down and to be removed from the cake pan. When it is removed from the pan, the cake will be ready for the last stage, which is decorating it with icing or any other eligible ingredients. Finally, when the decoration is over, the cake is ready to be served as dessert for house guests.

## **Open Provenance Model**

The OPM emerged as a result of the *Provenance Challenges* proposed in the context of IPAW. The *Provenance Challenges* came in four editions, one for each year from 2006 to 2010, except 2008, and OPM resulted from the first two challenges and was used on the third challenge:

*1st Challenge*: Aimed at providing 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 interoperability perspective.

The OPM is a provenance model designed to allow provenance information to be exchanged among systems using a common provenance model. It allows developers to build and share tools for such provenance model, while also supporting a digital representation of provenance for anything, whether it was generated digitally or not. Lastly, OPM defines a set of rules to identify valid inferences that can be done on provenance representations (MOREAU *et al.*, 2007).

OPM assumes that the provenance of objects is represented by a causality graph, which is a directed acyclic graph with annotations to capture more detailed information about the objects’ 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*”. Provenance, according to OPM, is composed of three types of elements (artifacts, processes, and agents) connected by causal relationships. The following sections will describe in details those aspects as well as the possibility of inferring provenance statements to simplify the provenance graph if necessary.

### 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. The definitions for all three vertices types are described in the following paragraph:

*“****Artifacts*** *are immutable pieces of state that can represent a physical object or its digital representation in a computer system.* ***Processes*** *are actions or a sequence of actions performed or caused by artifacts resulting in new artifacts.”* Lastly, *“****Agents*** *are contextual entities acting as a catalyst of a process that can enable, facilitate, control, or affect its execution”* (MOREAU *et al.*, 2007)*.*

# 

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

The edges of the provenance graph represent causal relationships between its source and destination. The edges in the provenance graph can belong to one of the five categories defined by MOREAU *et al.* (2007) and described in . Using the OPM notation, it is possible to generate a provenance graph from cake’s baking process, which is illustrated at Figure 2. In the cake example, the *artifacts* are the ingredients, the oven, and the cake. The *processes* are the mixing, baking, and decorating stages. Lastly, the *agent* is the cooker.

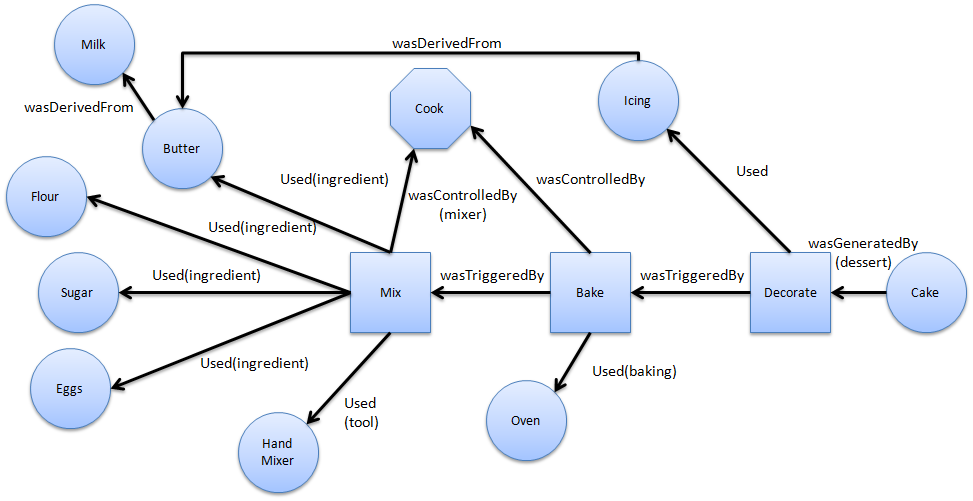


Figure 2: A simplified cake’s provenance graph.

A *used* edge that connects a *process* with an *artifact* indicates that the *process* required the *artifact* in order to complete its execution. If several *artifacts* are connected to the same *process*, then all of them were required for the *process* execution. It is also possible to specify each *artifact* role in the *process* execution by using the (*R)* field in the edge from , which represents role. Roles are not defined in OPM but are instead defined the application domain. Using the cake example, the *process* *mix* used butter, flour, sugar, eggs, and a hand mixer. Their roles in the *mix* *process* are ingredient (butter, flour, sugar, and eggs) and tool (hand mixer).

The edge *wasGeneratedBy* connecting an *artifact* with a *process* indicates that the *artifact* was only generated after the *process* started its execution. If several *processes* are connected with an *artifact*, then all these *process* must have started for the *artifact* to be generated. Similar to *used* edge, it is possible to specify the *artifact’s* role in the generation of the *process*. In the cake example, the *cake* was generated by the *process* *decorate* with the role of a *dessert* to be served to guests.

A *wasControlledBy* edge connecting a *process* to an *agent* indicates that the *agent* controlled the *process’s* start and ending. If the *process* was controlled by multiple *agents*, then their roles can also be represented in the in the relationship. Note however that this type of dependency represents a control relationship and not a data derivation. In the cake example, the *Cook* control the executions of the stages *mix*, *bake*, and *decorate*. He decides how long is the mixing, how long it takes to bake the cake, and how to decorate it.

The edge *wasTriggeredBy* connecting a *process* P2 with another *process* P1 indicates that the *process* P2 was only able to complete after the *process* P1 started. This edge also allows for a *process* to have an oriented view of past executions. Using the cake example, the *process* *bake* was only able to start after the *process* *mix* had started and, in this case, finished.

Lastly, the edge *wasDerivedFrom* connecting an *artifact* A2 with another *artifact* A1 indicates that the *artifact* A1 had to be generated for *artifact* A2 to be generated. This edge also gives an oriented dataflow view of the provenance. Using the cake example, the butter is made by churning milk. In other worlds, the milk is necessary for the creation of butter, so the butter was derived from the milk.

Moreover, OPMallows causality graphs to be used with time information. In OPM, time can be used to validate causality claims because if the same time clock is used to measure the passage of time for the effect and cause, then the effect will always come after the cause. In addition, time can be associated to *instantaneous occurrences* in a *process*. There are four types of occurrences (MOREAU *et al.*, 2007): *creation,* *use*, *starting* and *ending*. The first two are used for *artifacts* while the other two for *processes*. Because time is measured by someone, the model allows for a margin of error when representing time. For example, an *artifact* was used no earlier than time t1 and no later than time t2. This rationale is analogous for *processes*.

indicates that time information can be expressed in the model by using labels in the relationships. 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 mark when the process started and terminated. For *was derived from* and *was triggered by* edges, one timestamp to indicate when the *artifact* was generated. 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 is the cause for P2 execution.

### Inference

The OPMalso has defined the notion of a graph based on a set of syntactic rules and topological constraints (MOREAU *et al.*, 2007). The provenance graph captures causal dependencies that can be summarized by inferring facts, using transitive rules. Because of this, the OPMdefined a set of completion rules and inferences that can be used in the graph to improve understanding by omitting unnecessary steps. There are three completion rules: *artifact introduction*, *artifact* *elimination*, and *process introduction*.

For *artifact introduction*, Figure 3 illustrates the transformation of the *wasTriggeredBy* edge to *wasGeneratedBy* and *used* edges, introducing an *artifact* (cake batter). This is possible because the *wasTriggeredBy* edge is a composition of both edges. However, the identity of the *artifact* cannot be specified due to the lack of information, unless if the same *artifact* was previously in the graph and removed by inference or specified somewhere else. Analogous, the in *artifact elimination*, it is possible to transform the *used* and *wasGeneratedBy* edges to *wasTriggeredBy* edge, thus hiding the *artifact* (cake butter). This is possible because *wasTriggeredBy* edge is a composition of both edges.



Figure 3: Artifact introduction and elimination using the cake example.

The last rule is the *process introduction* illustrated by Figure 4, which is similar to *artifact introduction* but with a *process* instead of an *artifact*. While the *wasTriggeredBy* edge hides the presence of an *artifact*, the *wasDerivedFrom* edge hides the presence of an intermediary *process*. However, the *process introduction* rule can only be applied if butter is actually dependent of milk. This is the case, since in order to produce butter, it is necessary to churn the milk. So it is possible to add the *churning* *process* between milk and butter, thus introducing a *process*.

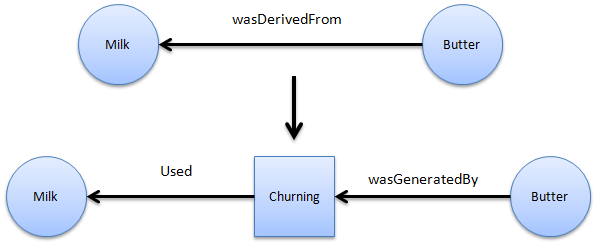


Figure 4: Process introduction in the cake example.

In a provenance graph, the causes of an *artifact* or a *process* might be traced to indirect causes from other *processes* and can involve multiple transitions between the cause and the perceived effect. For this purpose, OPM offers a set of relationships necessary for inferences: *Multi-step edges* (MOREAU *et al.*, 2007). The multi-step edges can be composed of four types: *wasDerivedFrom*\*, *used*\*, *wasGeneratedBy*\*, and *wasTriggeredBy*\*. Unlike their normal counterparts, these edges represent that multiple steps were necessary to be taken to have the same meaning as their normal counterparts.

Using the cake example, it is possible to make inferences by using multi-step edges, which are represented as red edges in Figure 5. The edge *wasDerivedFrom*\* says that the *artifact* *icing* was derived from *milk* (using a multi-step edge) indirectly, because the *butter*, which the *icing* is derived from, is derived from *milk*. This expresses that the *artifact* *milk* had an influence on *artifact* *icing*. Another possible multi-step is the *wasGeneratedBy*\* connecting *artifact* *cake* to *process* *mix*. This is possible because the *artifact* *cake* was generated by *processes* *decorate*, *bake*, and *mix*. The same can be said with *process* *decorate* *wasTriggeredBy*\* *process* *mix*, since *process* *decorated* could only have started after *process* *mix* had started, and ended in this example. Lastly, *process* *bake* *used*\* *artifact* *eggs* because *eggs* was necessary for the *process* *mix*, which in turn was necessary for *process* *bake* to start. As can be seen from these examples, multi-step edges can be used to infer single-step edges by eliminating *artifacts* and *processes* that occur in the chain of events.

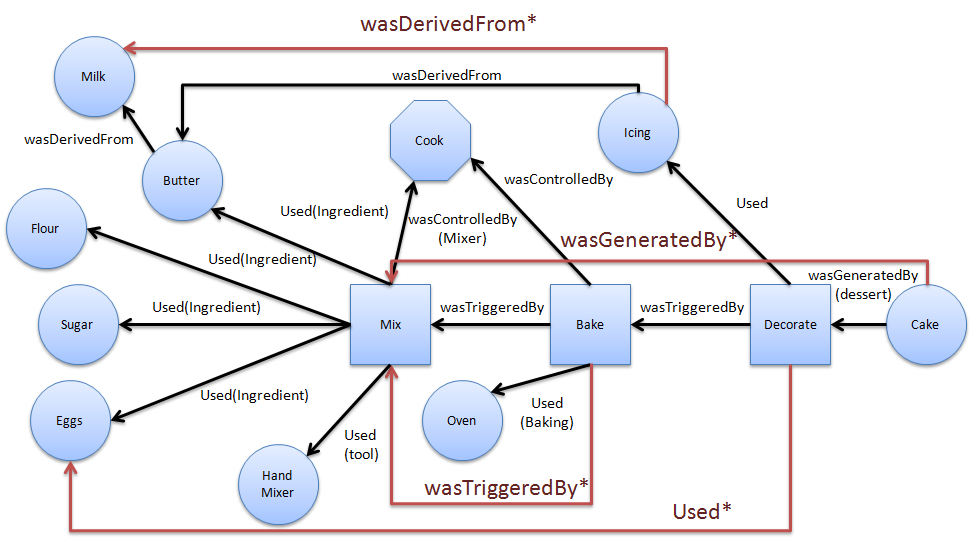


Figure 5: Multi-step edges in the cake example.

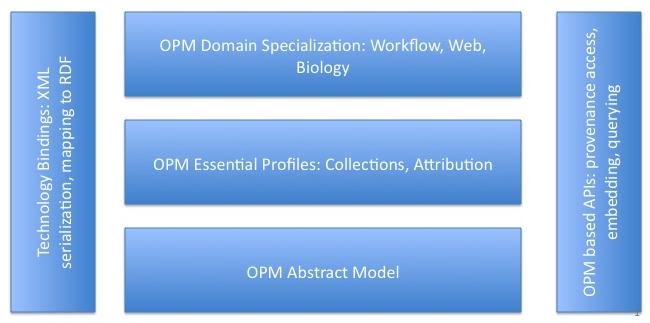


Figure 6: OPM’s Layered Architecture. Source: MOREAU *et al* (2007)

Lastly, the *Open Provenance Model* has a modular design as illustrated by Figure 6. However, specifications for all layers in the design have not been produced, possibly because the development team started working on another provenance model: PROV. Nevertheless, at the bottom layer is located the abstract model (MOREAU *et al.*, 2007). On the left hand side, is located a serialization to *xml*, defined by OPMX (The Open Provenance Model XML Schema) (MOREAU; GROTH; *et al.*, 2010), 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), and 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.

## **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 PROV, “*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*” (NIES *et al.*, 2010). The goal of this provenance model is similar to OPM’s, allowing provenance information to be exchanged among systems using a common provenance model while also enabling the representation by a provenance graph and using commonly used formats like XML (GROTH; MOREAU, 2010)..

The discussion group that developed PROV was officially launched concurrently to the forth *Provenance Challenge* (MILES *et al.*, 2010). The specifications that make up the PROV provenance model 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 7 illustrates the organization of PROV. At its core, is the conceptual data model that defines the common vocabulary used to describe provenance. Inside the data model, there is a set of constraints defined to aid developers in creating provenance programs to validate provenance statements. In order to support the interchange of provenance, PROV defined protocols to locate, access, and connect sets of provenance in order to aid their interoperability.

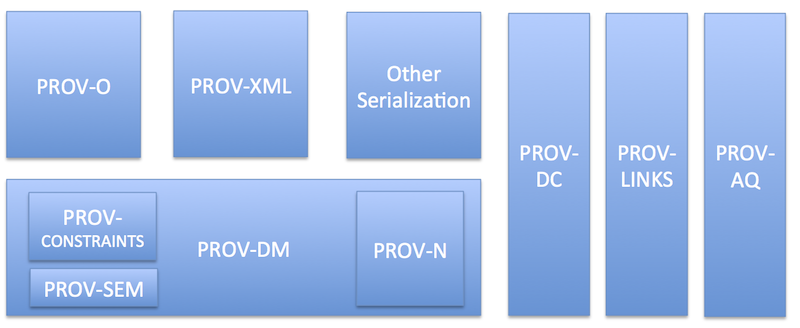


Figure 7: PROV organization. Source: (GROTH; MOREAU, 2010)

Provenance can be used for many purposes, from understanding how the data was collected in order to use it, to determine the object’s ownership, and decide if the information is trustworthy. It can also be used to check if the steps used in the process to obtain the result is compatible with the requirements. Lastly, it can be used to show the necessary steps to reproduce something. The PROV model specification accommodates these usages of provenance and provide three different ways to capture information according to the user’s perspective of provenance (GIL; MILES, 2010): Agent-centered, object-centered, and process-centered provenance.

The agent-centered approach focus on describing entities involved in the generation or manipulation of the information, while process-centered focus on capturing actions and steps used to generate the information. Lastly, the object-centered approach traces the origins of a document or artifact to other artifacts.

### Types and RELATIONS

PROV also uses a graph, similar to the provenance graph from OPM, to represent the provenance information. This graph is also characterized by having edges representing relationships between vertices and three types of vertices (GIL; MILES, 2010): *Entities*, *Activities*, and *Agents*.

Similar to *artifacts* from OPM, *entities* represents physical or digital objects like a document, the web, and material objects. *Activities*, which are similar to *processes* in OPM, are actions taken to change or interact with *entities* or *agents*. Lastly, an *agent* is a person, software, organization or *entities* that have responsibilities. This responsibility is a link to an activity. Several agents can have responsibilities to the same activity and a single agent can have responsibilities over several activities. Agents can also act on behalf of other agents, representing their interests when they are unavailable. These causal relations are some of the possible relationships available in PROV and are represented by edges in the provenance graph, similarly to OPM. Figure 8 illustrates those vertex 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):

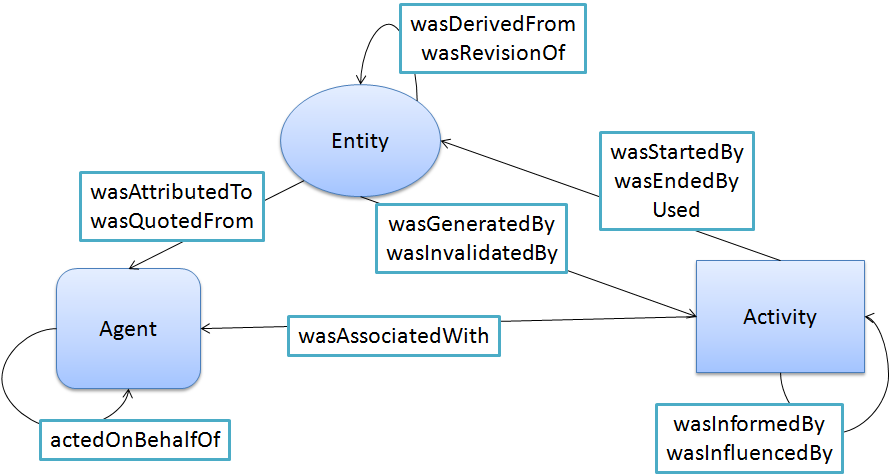


Figure 8: PROV Entities and relations.

Just like in OPM, the relationship denoted by an *used* edge from an *activity* to an *entity* indicates that the *activity* needed the *entity* for his operation. And before this occurrence, the *activity* did not require the *entity* and did not affect it unless otherwise specified by another relation. The relationship denoted by *wasStartedBy* edge from an *activity* to an *entity*, or between *activities*, indicates that the *entity* was the trigger that started the *activity*, similar to *wasTriggeredBy* from OPM. Similar to start, the wasEndedBy relationship indicates when the *activity* ended its execution from an *entity* or *activity*.

The relationships *wasGeneratedBy* and *wasDerivredFrom* are also similar to their OPM’s counterparts. The *wasInvalidadedBy* relationship between entities indicates that the entity is no longer available for usage. For example, in the cake scenario if the butter was home-made and there was only enough milk to make it for the cake and not the icing, then the milk was invalidaded by the butter, since there is no more left for future usage. The *wasInformedBy* edge indicates an exchange of an *entity* between two activities. In the cake example, it is unspecified the *cake batter* between *mix* and *bake*. The relationship *wasAttributedTo* from an *entity* to an *agent* indicates that the *entity* is now assigned to that *agent*, while *wasAssociatedTo* from an *activity* to an *agent* indicates that the *agent* had a responsibility or role in the *activity*.

The relationship between *agents* denoted by *actedOnBehalfOf* indicates that an *agent* assigned authority to another *agent* to execute *activities* on his behalf. The edge *wasRevisionOf* between *entities* indicates that the *entity* is a previous version of the other *entity*. The edge *wasQuotedFrom* indicates “*the repeat of an entity, such as text or image, by someone who may or may not be its original author*” (LEBO; SAHOO; MCGUINESS, 2010). Just like this definition quote from Lebo. Lastly, the *wasInfluencedBy* edge indicates that the *entity, activity* or *agent* affected another *entity, activity* or *agent* by influencing it.

### 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*, *new relations*, and *optional identifiers*. Their definitions, taken from MOREAU and MISSIER (2010a) are:

According to MOREAU and MISSIER (2010a), an specialization, denoted by *specializationOf* edge from an *entity* to another *entity* indicates that the first *entity* can do everything the second *entity* can, while also having more functions. Meanwhile, the edge *alternateOf* between *entities* indicates that both have the same functions and characteristics. Specialization and alternate are not considered in PROV as an influence. Subtyping is a rule to create new edges based in existing relationships, *entities* or *agents*. For example, chocolate cake is a special kind of cake. The revision relationship is also a subtyping of the derivation relationship, since a revision is a newer version based on the original document. PROV also supports the creation of totally new relationships without using any of the existing ones as a basis, which is useful depending on the domain used. For example, the relation *wasAttackedBy* is a relation between *agents* that indicates that one *agent* suffered an attack action from another *agent*.

The relationships represented by edges between two vertices can also be expanded to add more details. For example, in a revision relationship, it is possible to add further details about the changes between versions, describing how the entity was altered to generate the newer version. Another example is with plans used by agents. A plan, which in PROV can be represented as an entity, is a set of steps necessary to be taken in order to achieve the proposed goal. Figure 9 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).

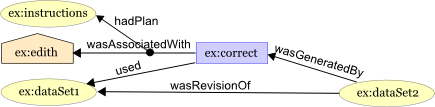


Figure 9: Using *Expanded Relations*. Source: GIL and MILES (2010)

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

Labels are used to name an *agent*, *entity* or *activity* for easier understanding, while locations are used to identify places that the agent, entity, or activity was located. Role, as previously mentioned, can be assigned to an *entity* or *agent* to provide an insight about their responsibilities during the execution of an *activity*. Type provides further information about the *agent*, *entity*, or *activity*. For example, *software agent* is a type of *agent*. Chocolate cake is a type of cake. Finally, value is used to represent values associated with the entity, which can be a string, a number, or encoded data. Table 1 describes which constructs are allowed the usage of attributes and if there is any restriction of its value.

Table 1: PROV optional attributes. Adapted from MOREAU and MISSIER (2010a)

|  |  |  |
| --- | --- | --- |
| 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.

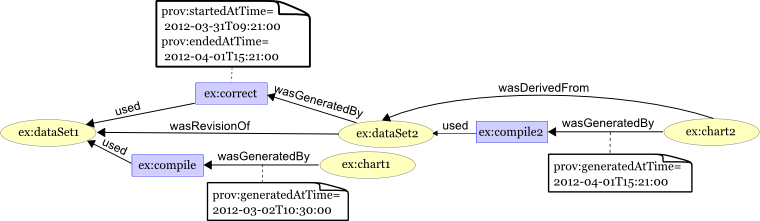


Figure 10: Time information. Source: GIL and MILES (2010)

### Inference

Like OPM, the PROV model also supports the usage of inferences on provenance data to identify indirect effects or influences, while also preserving the meaning. A set of rules to validate a provenance inference in PROV is available at (NIES *et al.*, 2010). An inference in PROV is a rule to add new statements or edges in order to simplify the information by skipping statements. It is also possible to replace statements/edges for equivalent ones to improve readability without changing the meaning of the statement (NIES *et al.*, 2010). For example, it is possible to change a derived relation to a revision relation if the meaning of the derivation in that situation was equal to a revision. This type of replacement in PROV is called as *definitions*. Both definitions and inferences can be viewed as logical formulas (NIES *et al.*, 2010). However, while *definition* can be used for both directions of an implication, this is not true for inferences. While looking at the activities to generate a cake (*mix*, *bake*, *decoration*), it is possible to infer that the cake was generated by the *activity* mix. However, from the perspective of the *activity* *mix* generating a cake, it is not possible to infer that *activities* *bake* and *decoration* were also involved.

## 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.

Table : OPM x PROV

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

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.

## 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.

MILES, Simon *et al.* *Provenance Challenge WIKI*. Available: <http://twiki.ipaw.info/bin/view/Challenge/>. Accessed: 26 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>.

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.