Troy Costa Kohwalter

<TÍTULO DO TRABALHO>

Escolher um item. Apresenta/da ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Escolher um item.. Área de Concentração: Escolher um item..

Advisors: Prof. Dr. Esteban G. W. Clua

Prof. Dr. Leonardo G. P. Murta

Niterói

2013

Ficha Catalográfica – Esta página deve ser removida na versão a ser entregue para a banca, mas deve ser reinserida na versão final, com a ficha catalográfica fornecida pela biblioteca. Informações sobre este processo devem ser obtidas na secretaria da pós-graduação.

Clique aqui para digitar texto.

Clique aqui para digitar texto.

Escolher um item. apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Escolher um item.. Área de Concentração: Escolher um item..

Aprovada em <MES> de <ANO>.

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Dr. <NOME DO ORIENTADOR> – Orientador

UFF

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Dr. <NOME DO AVALIADOR>

<INSTITUIÇÃO DO AVALIADOR>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Dr. <NOME DO AVALIADOR>

<INSTITUIÇÃO DO AVALIADOR>

Niterói

<ANO>

"Dedicatória(s): Elemento opcional onde o autor presta homenagem ou dedica seu trabalho" (ABNT, 2005).

**Agradecimentos**

"Elemento opcional, colocado após a dedicatória" (ABNT, 2005).

"Epígrafe: Folha onde o autor apresenta uma citação, seguida de indicação de autoria, relacionada com a matéria tratada no corpo do trabalho." (ABNT, 2005).

**Resumo**

"Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas e não de uma simples enumeração de tópicos, não ultrapassando 500 palavras" (ABNT, 2005).

Palavras-chave: "Palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores, conforme a ABNT NBR 6028" (ABNT, 2005).

**Abstract**

"Elemento obrigatório, em língua estrangeira, com as mesmas características do resumo em língua vernácula" (ABNT, 2005).

Keywords: "Palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores, na língua" (ABNT, 2005).

**Lista de Ilustrações**

[Figure 1: Edges in OPM. Source: (MOREAU *et al.*, 2007). 20](#_Toc353465411)

[Figure 2: Artifact introduction and elimination. Source: (MOREAU *et al.*, 2007). 23](#_Toc353465412)

[Figure 3: Process introduction. Source: (MOREAU *et al.*, 2007). 23](#_Toc353465413)

[Figure 4: Inference. Source: (MOREAU *et al.*, 2007). 24](#_Toc353465414)

[Figure 5: OPM’s Layered Architecture. Source: (MOREAU *et al.*, 2007) 24](#_Toc353465415)

[Figure 6: PROV organization. Source: (GROTH; MOREAU, 2010) 25](#_Toc353465416)

[Figure 7: PROV Entities and relations. Source: (GIL; MILES, 2010) 27](#_Toc353465417)

[Figure 8: Using *Expanded Relations*. Source: (GIL; MILES, 2010) 29](#_Toc353465418)

[Figure 9: *Optional ID* and *Further Relations*. Source: (GIL; MILES, 2010) 29](#_Toc353465419)

[Figure 10: Time information. Source: (GIL; MILES, 2010) 31](#_Toc353465420)

[Figure 11: Data model diagram. Gray classes represent provenance classes. 38](#_Toc353465421)

[Figure 12: Example of structure 40](#_Toc353465422)

[Figure 13: Relationship between a game using the framework and *Proof Viewer*. 42](#_Toc353465423)

[Figure 14: Example of a generated provenance graph. 43](#_Toc353465424)

[Figure 15: Collapsing Nodes 44](#_Toc353465425)

**Lista de tabelas**

[Table 1: PROV optional attributes 30](#_Toc353266335)

[Table 2: OPM x PROV 34](#_Toc353266336)

**Lista de abreviaturas e siglas**

"Elemento opcional, que consiste na relação alfabética das abreviaturas e siglas utilizadas no texto, seguidas das palavras ou expressões correspondentes grafadas por extenso. Recomenda-se a elaboração de lista própria para cada tipo" (ABNT, 2005).

**Lista de símbolos**

"Elemento opcional, que deve ser elaborado de acordo com a ordem apresentada no texto, com o devido significado" (ABNT, 2005).

Contents

[Chapter 1 – Introduction 16](#_Toc353465426)

[1.1 Motivation 16](#_Toc353465427)

[1.2 Goals 16](#_Toc353465428)

[1.3 Research Questions 16](#_Toc353465429)

[1.4 Contributions 16](#_Toc353465430)

[1.5 Organization 16](#_Toc353465431)

[Chapter 2 – Game Flow Analysis 17](#_Toc353465432)

[2.1 Introduction 17](#_Toc353465433)

[2.2 Usage 17](#_Toc353465434)

[2.3 UNDECIDED TITLE 17](#_Toc353465435)

[2.4 Related Work 17](#_Toc353465436)

[2.5 Final Considerations 17](#_Toc353465437)

[Chapter 3 – Provenance 18](#_Toc353465438)

[3.1 Introduction 18](#_Toc353465439)

[3.2 Open Provenance Model 19](#_Toc353465440)

[3.2.1 Types and Relations 20](#_Toc353465441)

[3.2.2 Time Information 21](#_Toc353465442)

[3.2.3 Completion Rules 22](#_Toc353465443)

[3.3 PROV 25](#_Toc353465444)

[3.3.1 Types and Notations 26](#_Toc353465445)

[3.3.2 Further Notations 28](#_Toc353465446)

[3.3.3 Time Information 30](#_Toc353465447)

[3.3.4 Inference 31](#_Toc353465448)

[3.4 Comparison Between Models 33](#_Toc353465449)

[3.5 Final Considerations 34](#_Toc353465450)

[Chapter 4 – Provenance in Games 35](#_Toc353465451)

[4.1 Introduction 35](#_Toc353465452)

[4.2 Data Model 36](#_Toc353465453)

[4.3 Provenance Gathering and Data Structure 39](#_Toc353465454)

[4.4 Provenance Visualization 41](#_Toc353465455)

[4.5 Final Considerations 45](#_Toc353465456)

[Chapter 5 – Implementation 46](#_Toc353465457)

[5.1 Introduction 46](#_Toc353465458)

[5.2 SDM 46](#_Toc353465459)

[5.3 Guiding Example 46](#_Toc353465460)

[5.4 Provenance Gathering 46](#_Toc353465461)

[5.4.1 Information Storage 46](#_Toc353465462)

[5.5 Provenance Analysis 46](#_Toc353465463)

[5.5.1 Node Representations 46](#_Toc353465464)

[5.5.2 Edge Representations 46](#_Toc353465465)

[5.5.3 Node Filters 46](#_Toc353465466)

[5.5.4 Edge Filters 46](#_Toc353465467)

[5.6 Final Considerations 46](#_Toc353465468)

[Chapter 6 – Evaluation 47](#_Toc353465469)

[6.1 Introduction 47](#_Toc353465470)

[6.2 Experiment Planning 47](#_Toc353465471)

[6.3 Experiment Execution 47](#_Toc353465472)

[6.4 Statistical Analysis 47](#_Toc353465473)

[6.5 Threats to Validity 47](#_Toc353465474)

[6.6 Final Considerations 47](#_Toc353465475)

[Chapter 7 – Conclusion 48](#_Toc353465476)

[7.1 Contributions 48](#_Toc353465477)

[7.2 Limitations 48](#_Toc353465478)

[7.3 Future Work 48](#_Toc353465479)

[References 49](#_Toc353465480)

[Glossary 53](#_Toc353465481)

[Appendix A – Título do Apêndice 54](#_Toc353465482)

[Annex A – Título do Anexo 55](#_Toc353465483)

[Index 56](#_Toc353465484)

# – Introduction

## Motivation

## **Goals**

## **Research Questions**

## Contributions

## Organization

# – Game Flow Analysis

## Introduction

## Usage

## **UNDECIDED TITLE**

## Related Work

## Final Considerations

# – 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.*, 2008a). 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).

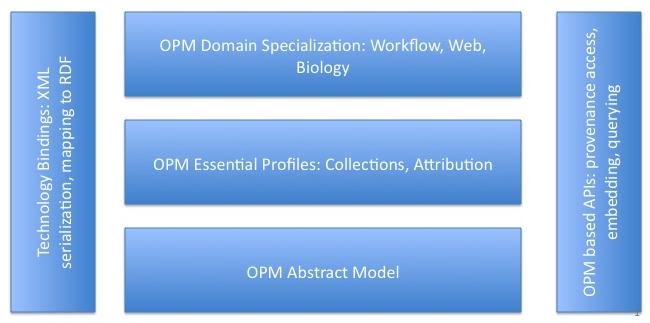


Figure 5: OPM’s Layered Architecture. Source: (MOREAU *et al.*, 2007)

## **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.

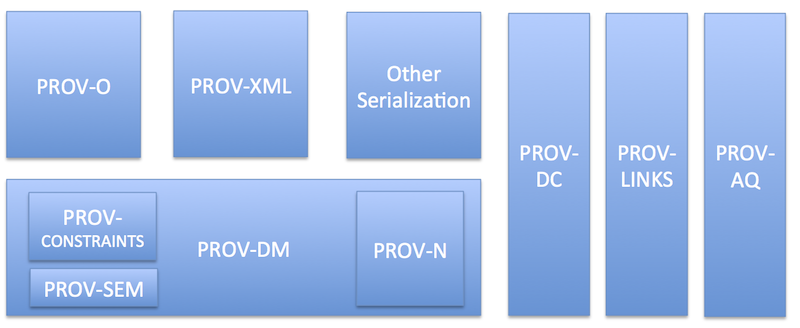


Figure 6: PROV organization. Source: (GROTH; MOREAU, 2010)

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):

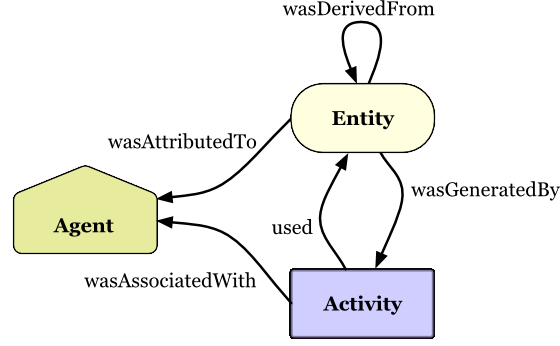


Figure 7: PROV Entities and relations. Source: (GIL; MILES, 2010)

**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).

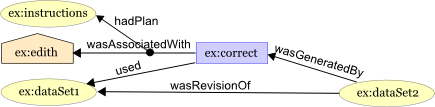


Figure 8: Using *Expanded Relations*. Source: (GIL; MILES, 2010)

***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”.

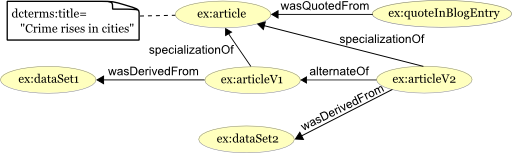


Figure 9: *Optional ID* and *Further Relations*. Source: (GIL; MILES, 2010)

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.

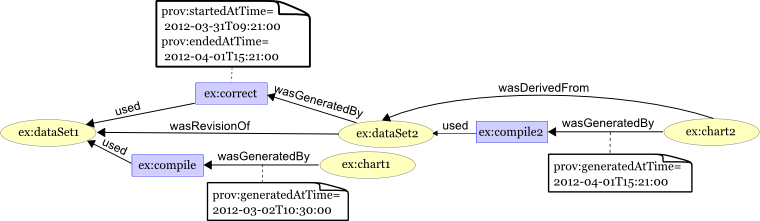


Figure 10: Time information. Source: (GIL; MILES, 2010)

### 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.

# – Provenance in Games

## Introduction

The conclusion of a game session derives from a series of decisions and actions made throughout the game. In many situations, analyzing and understanding the events, mistakes, and flows of a concrete game play may be useful for understanding the achieved results. A game flow analysis might be fundamental for detecting symptoms of problems that occurred due to wrong decision-making or even bad gameplay[[4]](#footnote-4) project. Without it, the player would be required to play the game again and make different decisions to intuitively guess which ones were not adequate to the situation. However, depending on the game dynamics and its complexity, reproducing the same state can be unviable, making it difficult to replay and try new solutions.

This game flow analysis deserve particular attention for serious games (ABT, 1987), which are games used for purposes other than entertainment while still providing pleasure. Serious games have been used for aiding students to learn and understand concepts taught in classrooms (BAKER; NAVARRO; VAN DER HOEK, 2003; NAVARRO, 2006) due to their stimulating curiosity characteristic and for providing motivation for learning (PRENSKY, 2001). Understanding the educational results obtained in a serious game is important to assimilate the knowledge and concepts passed in the game. In addition, examining the game flow allows for the identification of good and bad attitudes made by the player. This knowledge can be used in future game sessions to avoid making the same mistakes or even to adjust gameplay features.

Neural studies about the learning capability of human brain (CHIALVO; BAK, 1999; CLARK, 1950) state that the process of learning by correcting past mistakes is efficient and, consequently, desirable for the learning process. This process increases the human ability to adapt to new situations due to the rule of changing synaptic strengths, which ensures that synaptic changes occur only at neurons involved in wrong outputs. Nevertheless, in order to correct mistakes, it is fundamental to know which mistakes occurred.

As previously presented in Chapter 2, traditional games are limited in terms of analysis from the obtained results and as such, might compromise the player’s understanding of the story development throughout the game. Watching the game unfold again for a second time might not be enough to understand the reasons that affected the outcome, or how something happened the way it did, not the way it was expected to.

With this in mind, the goal of this work is to improve the player’s understanding of the game flow, providing insights on how the story progressed and influences in the outcome. In order to improve understanding, we provide the means to analyze the game flow by using provenance[[5]](#footnote-5). The provenance analysis is done by processing the collected gameplay data and generating a provenance graph, which relate the actions and events that occurred during the game session. This provenance graph allows the player, or a third user (ex: tutor), to identify critical actions that influenced the game outcome and helps to understand how events were generated and which decisions influenced them. This process also aids in the identification of mistakes, allowing the player to reflect upon them for future interactions.

In a previous work (KOHWALTER; CLUA; MURTA, 2012), I introduced the usage of digital provenance (FREIRE *et al.*, 2008b) in games. The main goal of that work was to propose a framework that collects information during a game session and maps it to provenance terms, providing the means for a post-game analysis. This was the first time that the provenance concept and formalization was used in the representation of a game flow. In this chapter it will be explained the *Provenance in Games* framework, along with the process of provenance gathering, the provenance graph construction and visualization.

## **Data Model**

In order to adopt provenance for the context of games, it is necessary to map each type of vertices of the provenance graph to elements that can be represented in games. As was mentioned in the previous chapter, the *Open Provenance Model* and PROV use three types of vertices: *Artifacts/Entities*, *Process/Activities* and *Agents*. In order to map these vertices types, it is first necessary to find their counterparts in a game context. To avoid misunderstanding, throughout this chapter it will be adopted the terms used in PROV for vertices (*entities*, *activities*, and *agents*).

Starting with *entities*, their provenance definition states that they are "*physical, digital, conceptual, or other kinds of things*". Its definition already gives a clue on which role they can represent in the game context: objects. An object can be anything used in the game, for example in the case of an RPG, *entities* can represent weapons, potions, legendary artifacts, magical objects, etc. It can represent anything meaningful to the development of the game history or even objects in a scene that someone interacted with.

On the other hand, *agent* definition is a "*person, a piece of software, an inanimate object, an organization, or other entities that may be ascribed responsibility*". In a game context, *agents* can be mapped as people represented in the game, non-playable characters (NPCs), monsters, and players. It can also be used to map event controllers, plot triggers or other entities involved in the story.

Lastly, *activities* according to its definition are "*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*". So, in a game context, *activities* can be viewed as actions or events made by living or intelligent entities that are present in the game. Note that it was made a difference between living and intelligent. This difference is important to mention because, for example, in an RPG environment a sword can be expressed as an agent because this sword has intelligence on its own. Despite being an object (sword), it can think and by an extent act, therefore it cannot be considered only as an object. It can also be as complex as being both an object and an agent at the same time.

Now, with all three types of vertices mapped into the game context, it is also necessary to map their causal relations to create the provenance graph. The PROV model defines some causal relations that can be used similarly to their original context, but can also be extended to be more suitable to the game context if necessary. Also, the PROV model can deal well with the aspect of time, which can be heavily explored in games, especially on games focused on storytelling, recording when each event happened and using this information to generate other events.

To generate actions and control events, each NPC in the game requires a behavior controller in order to control his actions, providing an array of behavior possibilities. It can be of any time, as long as the information is recorded when executed. Event triggers are also analogous. The information extracted will later be used in the provenance graph, so it is recommended to store relevant data.

Actions can be represented by a series of attributes that describe it and the context it was involved, allowing the creation of a provenance graph. As illustrated by , every action needs some information: a reason for its existence, why the action was performed, what triggered it, and who performed the action. In addition, the time of its occurrence can be important depending of the reason of using provenance. The main reason of using provenance, as discussed in this paper, is to produce a graph containing details that can be tracked to determine why something occurred the way it did. Therefore, with this assumption, the time of the action, the person who did it, what the action produced, and what it affect are recorded for further analysis.

For example, a monster attacked the player and scored a hit causing some damage, which in turns decreases the player’s hit points (HP). For this action, the relevant information is: when it was executed (time, turn, or combat round), who executed it (the monster), why it was executed (was it a special attack used because his HP was low? Or a normal attack?), who this action affected (the player), and the consequences of this action (decreased the player’s HP). If the action affects more than one person, then record all people involved and how the action affected each one. For example, that attack action was actually a buff attack, providing a boost the monster’s allies. So aside from recording the player and his damage, it should also be recorded the buff his allies received.

Events also work in a similar way as actions, with the difference in who triggered them, since events are not necessary tied to persons. For objects, its name, type, location, importance and the events that are generated by it can be stored to aid in the construction of the graph. Lastly, agents can have their names, attributes, goals, and current location recorded. illustrates this data model diagram, which maps provenance types to the context used in games. The gray classes are the original provenance objects used for the mapping. The diagram also exemplifies some basic information that should be considered for storage.

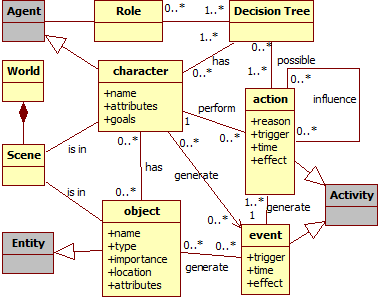


Figure 11: Data model diagram. Gray classes represent provenance classes.

## **Provenance Gathering and Data Structure**

In order to store all the necessary data to be used later for provenance reasons, it is required a storage structure. Depending on the information structure, it is possible to use the structure itself for inference in provenance, simplifying some unnecessary information.

Considering the generation of actions, which are executed by *agents*, it must be stored details about the action when it was executed. Example of the type of information to be storage from an action was given in the previous subsection and Figure 11. This action information can be stored in a structured similar to a list. This way, each *agent* will have a list of actions that contains all his executed actions. This structure allows for inferring the *agent* that executed each action by simply looking at whose list it belongs to, without the need to explicitly say who was responsible for the execution of the action. For events, it is possible to use an analogous approach, storing all events by trigger.

In the case of the action having an external influence from another action executed by another *agent* (for example, a buff spell), then the action will have a pointer to the action that influenced it. If an ally used a buff spell on me that buff my attack rate then, when my attack action is generated, it will be storage the action’s details and a pointer to my ally’s action that provided the buff. There will be no need to explicitly mention the ally, only his action because each action belongs to a list, which in turn belongs to an *agent*. With this structure, it is possible to infer who influenced the outcome of the action by following the links in each influenced action. If there were multiple influences in the executed action, then it is necessary to have a pointer to each action that influenced it. In the case of an action that generates influence, it is necessary to temporarily save a pointer to this action for the future actions that may be affected by it. Also indicate which type of action can be influenced by it.

For example, imagine a battle between a mage and an orc fighter. At the beginning of the battle, the mage cast a spell called “weakness” in the orc. This spell gives a penalty to the next attack roll. Because this action (spell “*weakness*”) generates an influence (in this case, a negative influence), it is necessary to save a pointer to this action to be used when the orc makes an attack action. Due to the distance between the orc and mage, the orc can’t make an attack action at this turn, so he charges in the direction of the mage to put him in melee range. On the next turn the mage cast another spell (“*stonefist*”), which causes only damage. In the orc’s turn, because he is now in melee range of the mage, he makes an attack action.

However, due to the spell casted by the mage (“*weakness*”) in the last turn, the orc suffers a penalty to his attack roll. In other words, the attack action from the orc was influenced by the spell action from the mage. When this happens, the attack action information that is stored will need to have a link to the spell action. Since a pointer to that particular action was already generated for future usage, then that pointer can be consumed (because the effect was only in the next attack) and used to link the actions. Without pre-generating the influence pointer, it would be necessary to check all lists in search of the action that influenced it. If the spell had a duration that affected all attacks until it expires, then it can be used a stack of pointers, where each turn one pointer will be consumed (but not necessarily bound to another action). When the stack empties, then that action cannot influence another because it expired. If the duration is not actually turns, but it is times used (the next five attacks), then each time an attack is made, it will consume one pointer from the stack and used to link actions. illustrates this combat scenario between the mage and the orc, including the influence between actions.

*Agents* present in a scene, or place, can also be represented in a similar way as actions. Each scene will have a list of *entities* and *agents* that belong to it. Analogous, a world is a list of scenes, which in turn contains a list of *entities* and *agents* that are in each scene. Each *agent* in turn has a list of performed actions, and each action will have links to other actions that influenced it. Using this structure, it is possible to simplify some inferences in the provenance model, such as to show only relevant actions, which has external influences, to evaluate the outcome of a game session. An example of such structure is also shown at , where the world has a list of scenes (Grasslands, Wasteland, Rock Island), each scene a list of *agents* (Mage and Orc), and lastly each *agent* has a list of performed actions (from their combat).



Figure 12: Example of structure

All these collected information made throughout the game, as they were executed, composes a game flow log that will be used later for the generation of a provenance graph. However, even using this storage information the game flow log can still be huge, which in turn will increase the size of the provenance graph. It is possible to make inferences in the provenance graph. However all information present in the graph is preserved even when inferences are made. An inference only omits information and does not remove them from the graph. So instead of recording everything in the game, applying a filter of which information is stored will decrease the size of the provenance graph. This filter can be done without losing any relevant information from the game.

For example, depending on the place, the number of *agents* present in it can be astronomical. Such is the case in cities or sometimes villages. So instead of collecting information from all *agents*, which most are there only walking around to give life to the city, it can be collected only from the ones that interacted with or influenced the actions of other *agents*. Doing this way, it will filter the *agents* that are only there to simulate a crowd. Another possible filter is for actions. For example, actions like sitting in a bench, opening a window, or jumping around while walking can be filtered. Filtering these types of non-essential actions or *agents* will decrease size of information gathered, which in turn will reduce the size of the provenance graph that will be generated later.

This filtering can also be done after the game flow log was generated or before it was used for the provenance graph. It can also be done in both stages, while the game was running and after the log was generated. While the game session is running, minor filters are used to reduce the game flow log size. And when the session is over, apply other types of filters to reduce even more the size of the log. The more irrelevant information is removed in this stage, it will be required fewer inferences during the graph visualization in order to clear the graph from unnecessary information. This way the user will be able to devote more of his attention to analyzing relevant data.

## Provenance Visualization

The purpose of collecting information during a game session is to be able to generate a provenance graph and use provenance techniques in order to analyze and infer the reasons of the outcome. In the previous sections, it was introduced the framework to store such information. However, not all stored information is relevant for the analysis. The provenance graph might contain actions that did not provoke any significant change in the game. These elements act as noise and can be omitted during provenance analysis by using completion and inference rules. With this in mind, this section introduces a novel provenance visualization tool named *Proof Viewer* (Provenance Flow Viewer), which is based on JUNG (JOSHUA O’MADADHAIN; DANYEL FISHER; TOM NELSON, 2010) and allows the analysis of generated game flow log through a graph. A game using the provenance in games framework is able to generate a game flow log that can be analyzed by *Proof Viewer*. Figure 13 illustrates the relationships between the game, using the framework, and Proof Viewer.

First, the game flow log, which contains game events, is processed and used to generate a provenance graph for analysis. After that, our tool creates the graph’s edges and nodes following our defined rules to generate the provenance graph. This graph is a representation of the game flow log and is available for the user to interact and analyze, reaching his own decisions about how events occurred during the game and the outcome. The user is able to see the consequences of each action and how they influenced other actions and the outcome, as well as manipulate the graph by omitting facts and collapsing chains of action for a better understanding and visualization process. No information is lost in this process, so the player can undo changes made during analysis.

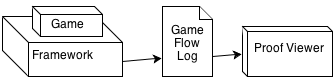


Figure 13: Relationship between a game using the framework and *Proof Viewer*.

A small example of a generated provenance graph from exported data is illustrated by Figure 14. Following the provenance notation specification, each node shape in Figure 14 is related to its type. Square nodes represent *process* nodes, circles are *artifacts* nodes and an octagon represents *agent* nodes as indicated by Figure 1. As can be seen in the figure, there is a chain of artifact nodes that represents the graph’s backbone. The edges in the provenance graph represent relationships between nodes, which can be *agents*, *artifacts* or *processes*. As such, *processes* nodes can be influenced positively or negatively by other *process* and have relationships with *artifacts* and *agents*. The context of such relationships may vary according to the type of relation between nodes.

Proof Viewer has other features besides node shape by type. It uses shapes and colors to distinguish displayed information and provides three types of filters: node filter, edge filter, and status filter. As previously noted, nodes have different shapes according to their types. However, it is also possible to differentiate a node from another with different borders and colors. As an example, *processes* that did not interact with other *processes* can be dotted, as illustrated in Figure 14. It is also possible to use different formats for edges. The thickness can be interpreted as how strong the relationship is. If the edge represents a low influence on the *process*, it is drawn as a thin edge. If the influence is high, then it becomes a thicker edge.

Another resource present for edges is color to represent the type of relationship. There are three types of relationship: positive, which indicates a beneficial relation; negative, which is a prejudicial relation; and neutral, which is neither beneficial nor prejudicial. For each type of relationship (positive, negative, and neutral) a different color is used. Green is used for positive influences, red for negative, and black for neutral. To emphasize the neutral relationships lack of importance, they are also dotted. These edge types are illustrated in Figure 14.

In order to better analyze graph data, the node filter feature is also available. Since the graph is generated from collected game date, not all collected information is relevant for every type of analysis. Thus, the provenance graph might contain actions that did not provoke any significant change. These elements act as noise and can be omitted during analysis. To do this, it is possible to collapse nodes in order to reduce the graph size by changing the information display scale, grouping nearby nodes together and thus changing the graph granularity. Another usage of collapse is to group *processes* from the same *agent*, making easier to see all influences and changes that the *agent* did throughout the game. Figure 15 illustrates a collapse by grouping an *agent’s processes* with the *agent*. Note the node size difference when collapsed. Another type of filter present is the edge filter. In the application it is also possible to filter edges by context and by the type of relationship.



Figure 14: Example of a generated provenance graph.

The last filter present is the status filter. When selecting the desired attribute, all nodes with the specified status will have their colors changed according to their respective values. It uses the traffic light scale (DIEHL, 2007), which indicates the status of the variable using red, yellow, or green color. As an example, imagine that we desire to analyze the player’s hit points (HP) value throughout the game. When filtered by player’s HP, all nodes that contain a player HP value will have their colors changed according to its value. Activating this type of filter allow the user to see the player’s HP throughout the game, making it easier to identify situations where he might have had trouble (red color). Section 5 provides more examples of those features used for analysis.

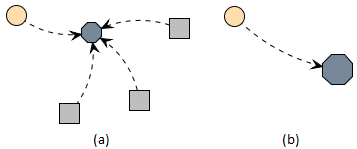


Figure 15: Collapsing Nodes

The normal graph, containing an artifact, an agent, and three processes (a) and the grouping of all agent processes with the agent (b).

Using these features for graph manipulation and visualization, the user is able to interact with the provenance graph, identifying relevant actions that had an impact in the story or in the desired type of analysis. It is also possible to hide information that might not be relevant to the desired analysis. Irrelevant information can be omitted in the graph or grouped together by features presented in the application.

With the aim of finding actions that had an impact in the story, the actions that did not cause any dramatic change can be omitted using multi-step inference rules. As an example, we may have a player in combat with an enemy and only after a few rounds it falls under the player's attacks. With the proposed framework, every round creates vertices to represent the actions taken by the player, which is attacking the enemy. This may generate data that is unnecessary for analysis, so it is possible to reduce all the individual attack nodes to simply one node. Another case could consist in a combat that does not generate any impact in the story outcome. In this case, it could be completely omitted.

However, that is not always true. The player could have made other actions against the enemy, which are also considered a form of attack, such as casting a spell, or a special attack maneuver, or even healing himself in order to survive. These actions are not duplicated, but can still be encapsulated for a superficial analysis and, if necessary, can be expanded for a detailed analysis. Note that all collected information is preserved and the only change made is on how it is displayed.

Since provenance is analyzed from the present to the past, the battle outcome is already known and can be used to decide if it was relevant or not. If the player was victorious with minor challenge, did not suffer severe wounds, or barely used any resources at his disposal, then the entire combat can be simplified into just one node representing that the player attacked the enemy and was victorious. However, if the combat was challenging or the player lost, it is interesting to show all action nodes for analysis, allowing the player to identify important facts that influenced the combat outcome.

Note however that Proof Viewer does not provide inference for the user, only the means necessary to infer. The player himself will need to decide which information is relevant for analysis. Providing a generic inference strategy is a future work. To infer something and decide if it is relevant or not for analysis is a complex process, which happens to be domain sensitive. This type of decision making also involve other areas of research (BRISTOL, 1977; CIOS; PEDRYCZ; SWINIARSKI, 1998; FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996; HAN; KAMBER, 2006; WITTEN; FRANK, 2005).

## Final Considerations

# – Implementation

## Introduction

In this work, the provenance in games framework is instantiated in the SDM game (KOHWALTER; CLUA; MURTA, 2011) as a proof of concept. The SDM game focuses on introducing Software Engineering concepts and skills to undergraduate students. The new and improved version of SDM presented in this paper includes provenance gathering and analysis, allowing students to visualize their actions and identify steps that lead to successful or unsuccessful outcomes. While the main application of provenance in this paper is over a serious game, we believe that the concepts discussed in this paper are applicable to other kinds of games and useful to support advanced analysis, such as gameplay balancing, events and behaviors data mining, and even storytelling enhancements.

## **SDM**

## **Guiding Example**

## Provenance Gathering

### Information Storage

## Provenance Analysis

### Node Representations

### Edge Representations

### Node Filters

### Edge Filters

## Final Considerations

# – Evaluation

## Introduction

## **Experiment Planning**

## **Experiment Execution**

## Statistical Analysis

## Threats to Validity

## Final Considerations

# – Conclusion

## Contributions

## **Limitations**

## **Future Work**

# References

ABT, Clark C. *Serious Games*. 1. ed. Abt Books: University Press of America, 1987.

BAKER, Alex; NAVARRO, Emily; VAN DER HOEK, André. Problems and Programmers: An Educational Software Engineering Card Game. *International Conference on Software Engineering(ICSE)*, p. 614–621, 2003.

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.

BRISTOL, Edgar H. Pattern recognition: An alternative to parameter identification in adaptive control. *Automatica*, v. 13, n. 2, p. 197–202, mar. 1977. Accessed: 25 nov. 2012.

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

CHIALVO, D.R.; BAK, P. Learning from mistakes. *Neuroscience*, v. v. 90, n. 4, p. 1137–1148, jun. 1999. Accessed: 5 jul. 2012.

CIOS, Krzysztof; PEDRYCZ, Witold; SWINIARSKI, Roman W. *Data mining methods for knowledge discovery*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

CLARK, George. The organization of behavior: A neuropsychological theory. *The Journal of Comparative Neurology*, v. v. 93, n. 3, p. 459–460, 1950. Accessed: 5 jul. 2012.

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.

DIEHL, Stephan. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. [S.l.]: Springer, 2007.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, v. 17, n. 3, p. 37, 15 mar. 1996. Accessed: 25 nov. 2012.

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

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

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.

HAN, Jiawei; KAMBER, Micheline. *Data Mining: Concepts and Techniques*. [S.l.]: Morgan Kaufmann, 2006.

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

JOSHUA O’MADADHAIN; DANYEL FISHER; TOM NELSON. *JUNG: Java Universal Network/Graph Framework*. sourceforge: Open-source, 2010. Available: <http://jung.sourceforge.net/>.

KOHWALTER, Troy; CLUA, Esteban; MURTA, Leonardo. Provenance in Games. In: 2012 XI BRAZILIAN SYMPOSIUM ON GAMES AND DIGITAL ENTERTAINMENT (SBGAMES), nov. 2012, Brasilia. *Anais*... Brasilia: In: XI SBGames, nov. 2012.

KOHWALTER, Troy; CLUA, Esteban; MURTA, Leonardo. SDM – An Educational Game for Software Engineering. In: 2011 X BRAZILIAN SYMPOSIUM ON GAMES AND DIGITAL ENTERTAINMENT (SBGAMES), nov. 2011, Salvador. *Anais*... Salvador: In: X SBGames, nov. 2011.

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

LINDLEY, Craig A. Narrative, Game Play, and Alternative Time Structures for Virtual Environments. In: GÖBEL, STEFAN *et al.* (Org.). *Technologies for Interactive Digital Storytelling and Entertainment*. Lecture Notes in Computer Science. [S.l.]: Springer Berlin Heidelberg, 2004. p. 183–194. Available: <http://link.springer.com/chapter/10.1007/978-3-540-27797-2\_25>. Accessed: 9 abr. 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.

NAVARRO, Emily. *Simse: a software engineering simulation environment for software process education*. 2006. California State University at Long Beach, Long Beach, CA, USA, 2006.

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

OXLAND, Kevin. *GAMEplay and Design*. [S.l.]: Addison-Wesley, 2004.

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>.

PRENSKY, Marc. Fun, Play and Games: What Makes Games Engaging. *Digital Game-Based Learning*, p. 1–31, 2001.

*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.

WITTEN, Ian H.; FRANK, Eibe. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. [S.l.]: Morgan Kaufmann, 2005.

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

# Glossary

"Elemento opcional, elaborado em ordem alfabética" (ABNT, 2005).

# Appendix A – Título do Apêndice

"Elemento opcional. O(s) apêndice(s) são identificados por letras maiúsculas consecutivas, travessão e pelos respectivos títulos. Excepcionalmente utilizam-se letras maiúsculas dobradas, na identificação, quando esgotadas as 23 letras do alfabeto" (ABNT, 2005).

# Annex A – Título do Anexo

"Elemento opcional. O(s) anexo(s) são identificados por letras maiúsculas consecutivas, travessão e pelos respectivos títulos. Excepcionalmente utilizam-se letras maiúsculas dobradas, na identificação dos anexos, quando esgotadas as 23 letras do alfabeto" (ABNT, 2005).

# Index

"Elemento opcional, elaborado conforme a ABNT NBR 6034" (ABNT, 2005).

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)
4. Gameplay is the pattern defined through the game rules, connection between player and the game, challenges and overcoming them, plot and player's connection with it (LINDLEY, 2004). It is the components that make up a rewarding, absorbing, challenging experience that compels player to return for more (OXLAND, 2004). [↑](#footnote-ref-4)
5. Provenance refers to the documented history of an object's life cycle and is generally used in the context of art, digital data, and science (PREMIS WORKING GROUP, 2005). [↑](#footnote-ref-5)