

---

# A Semantic Web Approach to the Provenance Challenge

Jennifer Golbeck<sup>\*,†</sup>, James Hendler<sup>‡</sup>

*MIND Lab, 8400 Baltimore Ave., Suite 200  
College Park, Maryland 20740 USA*

---

## SUMMARY

**Provenance is critically important for scientific workflow systems, as it allows users to verify data, repeat experiments, and discover dependencies. The Semantic Web is a natural fit for representing provenance, as it contains explicit support for representing and inferring connections between data and processes, as well as for adding annotations to data. In this article, we present a Semantic Web approach to the Provenance Challenge[MLA<sup>+</sup>07]. We use web services, ontologies, OWL reasoners, triple stores, and the SPARQL query language to implement the workflow, represent the data and the connections within it, and execute queries. We successfully implemented and answered all of the challenge queries. The flexibility of the Semantic Web also makes it quite easy to convert different provenance systems' data representation to a form we can work with. We illustrate this by integrating data from the PASS approach into our system, and successfully executing all of the challenge queries on it as well.**

KEY WORDS: *Semantic Web, Provenance*

## 1. INTRODUCTION

Provenance is critically important for scientific workflow systems, as it allows users to verify data, repeat experiments, and discover dependencies. In large distributed systems, such as interoperating web services and the Grid, provenance needs to represent the distributed execution and data storage successfully. Our approach to the Provenance Challenge[MLA<sup>+</sup>07] embraces the trend toward executing difficult scientific workflows in large distributed systems, by using the Semantic Web. A straightforward combination of basic Semantic Web technologies - ontologies, web services, reasoning, and rules - enabled us to successfully implement and answer all of the challenge queries.

The Semantic Web is an extension of the current web, designed to represent information in a machine-readable format. The web has evolved over the last fifteen years to incorporate technologies that make it easier for humans to understand the information being presented. However, this evolution

---

\*Correspondence to: MIND Lab, 8400 Baltimore Ave., Suite 200, College Park, Maryland 20740.

<sup>†</sup>E-mail: golbeck@cs.umd.edu

<sup>‡</sup>E-mail: hendler@cs.umd.edu

has also made it more difficult for computers to find much of the content on the web. While keyword-based text search works fairly well, searching for images, audio, video, or data is difficult and current technologies are not very effective. The Semantic Web addresses this problem by introducing knowledge representation languages based in XML. The Resource Description Framework (RDF), RDF Schema (RDFS), and the Web Ontology Language (OWL) are the languages of the Semantic Web, used for writing ontologies that describe data models and for encoding the data itself.

The Semantic Web is an ideal environment for representing provenance. The format of this data fits squarely in the paradigm of the Semantic Web. It is easy to represent annotations and connections between processes and files. Furthermore, the semantics of RDF and OWL allow us to define terms such that a reasoner can infer connections among files and processes. Given a specific file, all of the direct and indirect connections it has to other files and processes can be discovered with a simple query.

In this article, we present our approach to the provenance challenge using Semantic Web technologies. We begin with a discussion of implementing the workflow on the web using web services, and where provenance data is generated. We then describe how provenance for the files and processes is represented, and then present how we implemented all of the challenge queries. Because we claim that the Semantic Web is a natural fit for representing provenance, we illustrate that by choosing the most different approach to the challenge - Harvard's PASS system [SHBMR07] - performing a quick conversion of their data format into OWL, and showing that no modification to our system is necessary to query their data. Finally, we conclude with a discussion of ongoing work to personalize searches and filtering in provenance systems.

## 2. IMPLEMENTING THE PROVENANCE CHALLENGE

### 2.1. Executing the Challenge Workflow

Each of the initial files in the workflow are put online and assigned a URI. The URI is a unique identifier on the web. This URI allows us to reference the files in semantic web documents (see section 2.2), and to pass references to the files while we are executing the workflow.

Each process in the workflow is encapsulated in a web service. This allows the process to be invoked from anywhere, and to access files over the web. Each web service takes the URIs of the required files as inputs. The service then retrieves the actual files and executes the process, thereby creating the output files. These outputs are stored in a web-accessable environment, so each output file has a URI. The service also produces an OWL file describing the provenance information for its execution and for each output file (described in section 2.2). That OWL file is also stored on a web server, so it has a URI as well. When the service is finished executing, it returns its output: strings that contain the URI of each output file.

The entire workflow is executed through a web service (call this the Workflow Execution Service, WES). The inputs to the WES are the URIs of each of the initial files in the workflow (the Anatomy Headers, Anatomy Images, Reference Header, and Reference Image). That service, in turn, invokes the services representing the processes in the workflow. It passes in the URIs of the correct input files, and receives back the URIs of the output files. Those are passed to the next service and so on until the entire workflow has executed. The WES outputs the URIs of the final files, the three Atlas Graphics, and an OWL file describing its execution.

---

---

This is all that is necessary to have complete provenance for the execution of the workflow. To query the data, the OWL files must be aggregated into a system that stores the information, reasons over the data with the ontology, applies any rules, and supports queries. There are many systems that can handle all of these tasks. For convenience, we used the Redfood system which provides a web interface for browsing the data and executing rules and queries. Our entry can be viewed through this system at <http://provenance.mindswap.org>, but it is important to note that the details of Redfood are not part of the system entry; any semantic web system (or combination of systems) with basic OWL reasoning and rule support could just as easily be used to interact with the data.

## 2.2. Representing Provenance Information

In the previous section, we described the execution of the workflow, where many services produce OWL files describing their execution and output files' provenance. Representing information about provenance on the Semantic Web is relatively straightforward. We created an ontology for representing provenance.

An ontology defines classes of objects and properties that connect them. The data are instances of the classes. Data are connected to each other and to literal data via properties. A semantic web statement is in the form of a triple: subject predicate object. The subject is the datum being described, the predicate is the property of that datum, and the object is either a value for the property or another object connected to the subject by the property.

Given an ontology and the semantics of OWL, a system can make many logical inferences about the interconnectedness and dependencies of files and processes. For connections that did not directly follow from the semantics, we introduced Semantic Web rules to describe them. In this section we present an overview of the ontology, the rules, and the reasoning in the system.

### 2.2.1. Provenance Ontology

The ontology created to represent the data from the provenance challenge comprised several classes, and many properties, both arranged in a hierarchy. The ontology can be viewed online or loaded into any semantic web tools for browsing: <http://provenance.mindswap.org/provenance.owl>

This ontology was designed to represent the workflow of the Provenance Challenge. Its features are designed around that task, and as a result it is not a totally generic ontology. While it could be applied to other domains, it is likely that the designers of other applications will want to extend or modify the ontology to represent their view of their system.

Our top level class is a Resource, under which we described classes for Files (and their subtypes), web services, and the execution of the entire workflow and of individual web services (which were encapsulations of the processes). The classes are connected with properties.

The properties in the ontology reflect what data we recorded for each class of objects. We had no properties with a domain of the WebService class, because the OWL-S project is an extensive ontology for representing information about web services. For the classes File, ServiceExecution, and WorkflowExecution, we defined several properties. Below we describe features of those classes, and the properties for each.

---

**ServiceExecution:** The class `ServiceExecution` describes a specific execution of a service. It is used to record the time, date, inputs and outputs of that specific invocation. We created properties for each of the following data.

- **Input and Output files:** For each instance of a service execution, we have a property that points to the URIs of all the input files and all the output files.
- **Input parameters:** Some services required input parameters, such as the “-m 12 -q” parameters passed to executions of `align_warp`. This property uses a string to indicate them.
- **Date and time:** Because the timing of an execution is important provenance, and because it is included in some of the challenge queries, we included several properties indicating the date, time, and day of week of the service execution.
- **Connection to workflow execution:** Each service execution is linked back to the description of the workflow execution that invoked it.
- **Service used:** This property links a service execution to the description of the web service that was executed.
- **Stage:** Although the stage number was only a label in the description of the challenge, it was required for some of the queries. Thus, we include a property to annotate each service execution with the stage it is the workflow.

**File:** The class `File` is used to represent provenance for all the files produced by the workflow. The properties we included are used to record simple literal data as well as to connect files to other files and to `ServiceExecution` instances.

- **File parents and file ancestors:** When a file is created, the input files that were used to produce it are its file parents. File ancestry connects a file to every file that it is directly or indirectly dependent on in the workflow execution. We defined a file ancestry property as transitive, which means that if a file *X* has an ancestor file *Y*, all of file *Y*'s ancestors are also file *X*'s ancestors. Transitivity is part of the semantics of OWL, so a reasoner will infer ancestry between two files connected by a chain of ancestors. This is a critical component of the ontological approach to the challenge. A sub-property of file ancestor is file parent. A file parent is any file that was directly used as input for the file being described. While parenthood is not transitive, the fact that it is a subproperty of ancestry means that a chain of files with parent links is *also* a chain with ancestry links, and thus ancestry can be inferred through parenthood.
- **Produced by Service:** This property connects a `File` to the web service that created it. This connection is not to the specific execution of that service, but rather a link to the process that was used to create it.
- **Service Execution Parents:** Each file is connected to the service execution that produced it. The property that makes this connection is the inverse of the property that connects service executions to their output files. Service execution ancestry is also important, as it ties each file to all of the service invocations in its history. However, the semantics of OWL alone are not sufficient to represent this type of role composition. To infer service execution ancestry, we have a rule, described in section 2.2.2.
- **Siblings:** Files that were output by the same web service are considered output siblings. A property is used to connect all files output by the same service execution

- Date and Time information: We have a property for indicating the date and time that a file was created.

**Workflow Execution:** Finally, we represented information about the execution of the entire workflow. This allows us to connect all of the service executions and files to a specific workflow.

- Service Executions: we have a property to connect the workflow to each service execution that it invoked. This property is the inverse of the service execution's connection to the workflow.
- Times: We record the start and end times of the workflow execution.

### 2.2.2. Semantic Web Rules

In addition to the properties contained in the ontology, we needed to know ancestry for service executions, i.e. the ServiceExecution used to produce the file is the ServiceExecutionAncestor of all that file's descendants. We encoded this as a Semantic Web rule that establishes a ServiceExecutionAncestor relationship to a file if the file has an ancestor file produced from a specific ServiceExecution. This rule (and any others we may choose to add) are then applied to the knowledge base using Pchinko, a Semantic Web rule engine[KCP05].

## 2.3. Implementing Challenge Queries

Using the representation, logic, and rules described in section 2.2, we were able to successfully answer all nine challenge queries. The queries were formulated using SPARQL, and the results could be returned in a variety of formats.

SPARQL[PS06] is the SPARQL Protocol And RDF Query Language, a World Wide Web Consortium (W3C) initiative. One of its major components is the SPARQL Query Language which provides a syntax for querying RDF graphs. A query is structured with three main parts: the prefixes, the variables being selected, and the structure of the triples. Consider the following query, corresponds to query 1 of the challenge.

```
PREFIX workflow: <http://provenance.mindswap.org/files/ConvertExecution-1155932184.62982.owl#>
SELECT DISTINCT ?property ?value WHERE {
workflow:Graphic1155932184.62982 ?property ?value.
}
```

The prefix line allows us to define a prefix to use as shorthand in the query. Instead of including the full URI every time we refer to an object, we can replace the first part of the URI with its prefix. In this example, the file that contains all of the data for the workflow is assigned the prefix "workflow". The second line uses SQL-like syntax for listing the variables we want to select. Finally, the body of the query gives a pattern of triples that must be matched. Our first query requests all the properties and values of the Atlas X Graphic, represented by its URI with a prefix.

SPARQL also has a facility to filter what values may be used with a variable. In the next example, which satisfies query 2, we want to find all information about the Graphic excluding everything before the application of softmean. Thus, we filter the data to show only processes (ServiceExecutions) and their data where softmean is the service used by ServiceExecutionAncestor (this includes direct parents, because of the property hierarchy). If softmean is not the service of an ancestor, then it is not in the history of the file or process, and thus must not have yet occurred.

---

```

PREFIX prov: <http://provenance.mindswap.org/provenance.owl#>
PREFIX workflow: <http://provenance.mindswap.org/files/ConvertExecution-1155932184.62982.owl#>
SELECT DISTINCT ?subject ?property ?object WHERE
{
  ?subject ?property ?object.
  workflow:Graphic1155932184.62982 ?prop1 ?subject
    FILTER (?prop1 = prov:hasServiceExecutionAncestor).
  ?subject prov:hasOutputFile ?file.
  ?file ?prop2 ?service
    FILTER ( ?prop2 = prov:hasServiceExecutionAncestor).
  ?service prov:serviceUsed prov:softmean.
}

```

Filtering can also be more advanced. The following query satisfies query 9 of the challenge. Here, we use compound conditions in the filter to allow for any of the annotations that the challenge query specified.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX prov: <http://provenance.mindswap.org/provenance.owl#>
SELECT DISTINCT ?graphic ?annotation WHERE {
  ?graphic rdf:type prov:Graphic.
  ?graphic prov:annotation ?annotation.
  ?graphic prov:annotation ?oneOfThese
    FILTER ( ?oneOfThese = "studyModality=speech" ||
             ?oneOfThese = "studyModality=audio" ||
             ?oneOfThese = "studyModality=visual" ).
}

```

These queries can be run in any system with SPARQL support. We run them in our web interface, as shown in figure 1. Our challenge entry website<sup>†</sup> contains the SPARQL for each of the queries. We were able to successfully implement and answer all nine queries in the challenge.

### 3. COMPARISON WITH OTHER APPROACHES

The benefit of the Semantic Web approach is that any data from any sources can be integrated into the knowledge base. For example, we could use data from any of the other approaches to the challenge that rely on relational databases by quickly creating an ontology representation of the database schemas, and producing a corresponding RDF file. The work required to do this is minimal; there are existing tools (e.g. [GML05]) that will convert database schemas to Semantic Web formats. Similarly, any of the data from challenge participants who use XML could quickly be included in our knowledge base with the addition of an ontology.

Several other teams used OWL or RDF representations for their data. The USC/ISI approach [KDG<sup>+</sup>07] used OWL to create ontologies and as a format for their data. Furthermore, like us they

---

<sup>†</sup><http://twiki.ipaw.info/bin/view/Challenge/MINDSWAP>

Figure 1. The output of query 2 when run in our web interface. Each instance can be expanded to see all its properties and values.

used SPARQL to query the provenance knowledge base. Other teams were similar in their formats for data representation, including approaches from NCSA [FM07] and myGrid from University of Manchester [ZGST07], both using RDF. As is to be expected, the underlying knowledge models are different in each approach, just as one would expect the database schemas to be different among teams storing data that way. The major difference between our full solution to the challenge and all the other teams, is that our approach was fully web based. We did not have a workflow system as nearly every other team did. Instead, each step in the workflow was represented as a single service that executed on the web and produced a stand-alone provenance file. Thus, while the representation of the data in Semantic Web format offers many benefits, our system is unique in the distributed, independent nature of the procedures.

While several teams chose to use Semantic Web formats for their data, basically all formats can be converted into an RDF representation. Then, our browsing and querying front end can be used to interact with that data. To illustrate this, we chose the system that was most different from ours: Harvard's PASS [SHBMR07] where provenance is part of the file system.

We obtained a full dump of the PASS database and converted it into OWL. We mapped processes to ServiceExecutions and files to instances of the File class. The properties from the database were mapped, when possible, to the corresponding properties in our ontology. For example, the PASS INPUT\_FILE property maps to the hasInputFile property if the PASS pnode is a process, or to the hasServiceExecutionParent property if the pnode is a file. We also added a rule to link files with the

---

---

input files used to create them. This was recorded directly in our workflow execution, but needed to be inferred from the PASS data.

Once this conversion has been made and the data is included in our triple store, we can execute queries on it in the same way as we did for our own data. Some queries can be run with no modifications at all. The SPARQL for query 1 only requires substituting in the URI of the Atlas X graphic produced by PASS for the URI from our workflow execution. Other queries require minor modifications due to representational differences. For example, query 8 needs modification for finding Service Executions using `align_warp` because the PASS data indicates the process used by its system path while the MINDSWAP system uses a URI. A more complex conversion could create URI representations of these paths, so only URI changes would be necessary. As it stands, though, we were able to successfully execute all of the queries on the converted PASS data with only minor modifications from the syntax used on the MINDSWAP data. This is a benefit of Semantic Web systems, which do not require the precise mapping needed for converting between database schemas. Furthermore, when data is not included in one representation (such as the direct connection between processes and their outputs in PASS), it can often be logically inferred in a Semantic Web system.

#### 4. CONCLUSIONS AND FUTURE WORK

In this article, we presented our Semantic Web technique for the provenance challenge. Using ontologies, web services, and the SPARQL query language, we were able to successfully implement the challenge workflow and answer all the queries. We also showed that with a relatively straightforward conversion of data into OWL - even if the original source of that data is very different from our own - we can successfully answer the challenge queries with very little modification.

We are in the process of integrating social networks and trust into our system. Our previous work has presented methods for computing trust between people in social networks[Gol05]. If a provenance system tracks who created files or executed the processes in a workflow, then the trustworthiness of that person can be used as a measure of the quality of the results. We are investigating methods for filtering results based on trust and how to use trust information when there are multiple sources for a file. We believe social filters on scientific data are common in real life when people judge data quality, and that an automated mechanism for using it in a provenance system would be interesting and useful.

#### ACKNOWLEDGEMENTS

This work, conducted at the Maryland Information and Network Dynamics Laboratory Semantic Web Agents Project, was funded by Fujitsu Laboratories of America – College Park, Lockheed Martin Advanced Technology Laboratory, NTT Corp., Kevric Corp., SAIC, the National Science Foundation, the National Geospatial-Intelligence Agency, DARPA, US Army Research Laboratory, NIST, and other DoD sources.

#### REFERENCES

- [FM07] . Joe Futrelle and Jim Myers. Tracking provenance semantics in heterogeneous execution systems. *Concurrency and Computation: Practice and Experience*, 2007.

- 
- [GML05] . John H. Gennari, Peter Mork, and Hao Li. Knowledge transformations between frame systems and rdb systems. In *K-CAP '05: Proceedings of the 3rd international conference on Knowledge capture*, pages 197–198, New York, NY, USA, 2005. ACM Press.
- [Gol05] . Jennifer Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland, College Park, 2005.
- [KCP05] . Yarden Katz, Kendall Clark, and Bijan Parsia. Pychinko: a native python rule engine. In *Proceedings of the International Python Conference*, 2005.
- [KDG<sup>+</sup>07] . Jihie Kim, Ewa Deelman, Yolanda Gil, Gaurang Mehta, and Varun Ratnakar. Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 2007.
- [MLA<sup>+</sup>07] . Luc Moreau, Bertram Ludäscher, Ilkay Altintas, Roger S. Barga, Shawn Bowers, Steven Callahan, George Chin Jr., Ben Clifford, Shirley Cohen, Sarah Cohen-Boulakia, Susan Davidson, Ewa Deelman, Luciano Digiampietri, Ian Foster, Juliana Freire, James Frew, Joe Futrelle, Tara Gibson, Yolanda Gil, Carole Goble, Jennifer Golbeck, Paul Groth, David A. Holland, Sheng Jiang, Jihie Kim, David Koop, Ales Krenek, Timothy McPhillips, Gaurang Mehta, Simon Miles, Dominic Metzger, Steve Munroe, Jim Myers, Beth Plale, Norbert Podhorszki, Varun Ratnakar, Emanuele Santos, Carlos Scheidegger, Karen Schuchardt, Margo Seltzer, Yogesh L. Simmhan, Claudio Silva, Peter Slaughter, Eric Stephan, Robert Stevens, Daniele Turi, Huy Vo, Mike Wilde, Jun Zhao, and Yong Zhao. The First Provenance Challenge. *Concurrency and Computation: Practice and Experience*, in this issue, 2007.
- [PS06] . Eric Prudhommeaux and Andy Seaborne. Sparql query language for rdf. In <http://www.w3.org/TR/rdf-sparql-query/>, 2006.
- [SHBMR07]. Margo Seltzer, David A. Holland, Uri Braun, and Kiran-Kumar Muniswamy-Reddy. Pass-ing the provenance challenge. *Concurrency and Computation: Practice and Experience*, 2007.
- [ZGST07] . Jun Zhao, Carole Goble, Robert Stevens, and Daniele Turi. Mining taverna’s semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 2007.