

# On the Use of Abstract Workflows to Capture Scientific Process Provenance

Paulo Pinheiro da Silva  
*Computer Science Department*  
*University of Texas at El Paso*  
*El Paso, Texas 79968*

Nicholas Del Rio  
*Computer Science Department*  
*University of Texas at El Paso*  
*El Paso, Texas 79968*

Leonardo Salayandia  
*Computer Science Department*  
*University of Texas at El Paso*  
*El Paso, Texas 79968*

Ann Q. Gates  
*Computer Science Department*  
*University of Texas at El Paso*  
*El Paso, Texas 79968*

## Abstract

Capturing provenance about artifacts produced by distributed scientific processes is a challenging task. For example, one approach to facilitate the execution of a scientific process in distributed environments is to break down the process into components and to create workflow specifications to orchestrate the execution of these components. However, capturing provenance in such an environment, even with the guidance of orchestration logic, is difficult because of important details that may be hidden by the component abstractions. In this paper, we show how to use abstract workflows to systematically enhance scientific processes to capture provenance at appropriate levels of detail. Abstract workflows lack the orchestration logic to execute a scientific process, and instead, are intended to document scientific processes as understood by scientists and at the level of detail that is necessary to capture provenance for scientific use. Furthermore, abstract workflows are coupled with a representation of provenance that can encode provenance generated from distributed components. We show how the approach described in this paper has been used for capturing provenance for scientific processes in the Earth science, environmental science, and solar physics domains.

## 1 Introduction

Most scientists capture provenance about their data and process executions without using methods specifically designed to record provenance. For example, scientists may record provenance in the form of logs, meta data, annotations, and others. However, it is often hard for other scientists to reuse provenance recorded in such forms. For example, scientists may need to delve into the inner-workings of a software system to understand how a log is being constructed and to determine whether the information provided in the log is adequate to sup-

port provenance-related analysis. A common representation for provenance, i.e., a provenance language, has been identified as a necessity to facilitate the reuse of provenance. This language should be used to capture provenance for the relevant parts of the scientific process, whether the process is executed within a machine, executed in a distributed environment, or even if the process is not executed in an electronic way, i.e., human activity. We use the Proof Markup Language [9] as our preferred domain-independent language for encoding provenance.

Furthermore, to extend scientist adoption of methods specifically designed to capture provenance, we discuss the issue of supporting scientists to systematically enhance their processes to capture provenance whether these processes are specified as scientific workflows or not, whether they are centralized or distributed, and whether they are executed by machines or carried out manually by humans. The use of abstract workflows [2] has been developed to facilitate the sharing of process knowledge (as defined in [3]) among scientists. Abstract workflows are specifications that are not committed to be executed by machines; hence, abstract workflows have the added flexibility of being useful to describe distributed scientific processes that are hosted across cyber-environments regardless of the specific technologies employed. This paper describes how abstract workflows are used to generate wrapper modules to enhance distributed scientific processes with functionality for receiving, capturing, and propagating provenance. These modules are called data provenance annotators. The paper also describes how abstract workflows support the capture of provenance for processes that are carried out by humans.

The rest of the paper is organized as follows. Section 2 introduces a scientific process use case for distributed provenance. Section 3 describes the languages and tools used for capturing distributed provenance for the scientific process use case. Section 4 describes the provenance capturing approach. Section 5 reports on other efforts where the provenance capturing approach has been used.

Section 6 describes approaches for capturing provenance used in other computational platforms. Conclusions are presented in Section 7.

## 2 Use Case - Hole's Code

To illustrate how our techniques can be integrated into complex workflows, we will refer to a process that creates a seismic velocity model of the Earth's crust using a non-linear tomography inversion procedure [4], a finite difference calculation [11], and observed velocities of seismic waves through structures of the Earth's crust. The process, referred to as Hole's Code, is illustrated in Figure 1. The process starts by obtaining a preliminary velocity model of the Earth's crust that is constructed from field measurement data (*Vel1-3D*). Next, the velocity model is refined gradually by executing an iterative process that uses seismic waves generated from controlled shot-point explosions (*Punch*) and corresponding measurements of the arrival times of the waves at geophone stations (*Cover*). The last three steps of the iteration correspond to a smoothing step (*Tomo*), a filtering step (*Duadd*), and a step to incorporate the refinements to the velocity model (*Addc*). Such models are useful for earthquake analysis and oil exploration.

## 3 Background

### 3.1 Ontologies and Abstract Workflows

It is important for scientists to document the scientific processes that they use to generate scientific artifacts. It is also important for other scientists to be able to understand the processes that were used to produce scientific artifacts. Scientific workflows is one approach to document scientific processes.

By using technologies from the Semantic Web community, as well as elemental principles from software engineering practices, the CI-Miner approach [2] provides a technique to document scientific workflows that is amenable to users of non-technical fields of expertise. The approach consists of creating task ontologies to capture domain knowledge that effectively represents a controlled vocabulary of a project, as well as additional knowledge that suggests the use of this vocabulary towards the description of processes for that project. Next, the scientist uses the knowledge encoded in the ontology to document scientific processes in the form of abstract workflows. The ontologies used in this approach are referred to as Workflow-Driven Ontologies (WDOs) [10], and the WDO-based workflow specifications are referred to as Semantic Abstract Workflows (SAWs). Both are discussed further in the next subsections, as well as the WDO-It! tool that assists the user through the approach.

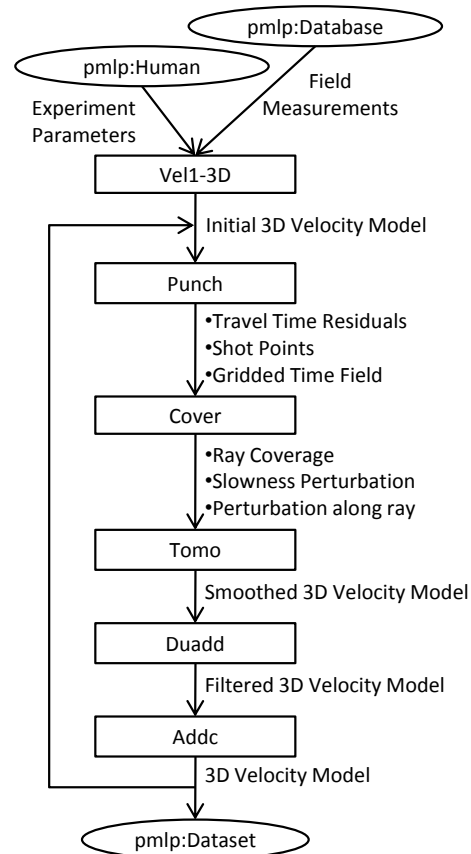


Figure 1: Semantic Abstract Workflow for Hole's Code

### 3.1.1 Workflow-Driven Ontologies

Guarino [3] suggested the classification of ontologies according to their level of dependence to a particular task or point of view. In this classification, WDOs are task ontologies; WDOs are encoded in OWL and are used to document concepts about a domain for the purposes of capturing process knowledge.

The two main classes of WDOs are *Data* and *Method*. The *Data* class is representative of the data components of the scientific process. These can be things such as datasets, documents, instrument readings, input parameters, maps, and graphs. The *Method* class is representative of discrete activities involved in the scientific process that transform the data components. As described in [10], the intention of WDOs is to allow scientists to capture process-related classes by extending the hierarchies of *Data* and *Method*.

### 3.1.2 Semantic Abstract Workflows (SAW)

From the perspective of an end-user, scientific workflows can be generalized as graphical structures that contain nodes representing discrete activities and directed edges representing data flow between those activities. Activities connected through directed edges effectively determine data dependencies between the activities. Traversing the graph from its initial data sources to its final data sinks simulates the action of carrying out a complex process conformed of simpler activities. To deploy such a representation of a process as an automated or semi-automated system, additional control flow information is necessary to determine the rules that guide the graph traversal.

According to the CI-Miner approach, the main artifact for scientists to capture scientific processes is a SAW. Semantic refers to the meaning inherited by using ontological classes captured in a WDO. Abstract refers to the fact that the workflows captured lack the additional constructs necessary to produce automated systems that would implement the modeled workflow. In this sense, SAWs are not committed to be executable workflow specifications.

Figure 1 shows an example of the graphical notation of SAWs. Data are represented by directed edges and Methods are represented by rectangles. Data and Methods are labeled with the name of their corresponding user-defined WDO class. Sources and Sinks are introduced in the graphical notation of SAWs as a bootstrapping mechanism to indicate the starting and ending points of a process, and these are represented by ovals. Sources and Sinks are also labeled with the name of their corresponding class defined in the provenance component of the Proof Markup Language (PML-P) ontology discussed below.

### 3.1.3 WDO-It!

In terms of the Semantic Web community, and the OWL language in particular, WDOs are OWL documents that capture ontological classes and relations, while SAWs are OWL documents that capture knowledge bases based on the knowledge captured in the WDOs. Hence SAWs do not contain class or property definitions, but instead, include only instances of the classes and properties defined in WDOs.

WDO-It! (<http://trust.utep.edu/wdo>) is a Java-based tool intended to help scientists in the creation of WDOs and SAWs encoded in OWL. In order to document scientific processes with SAWs, a scientist would start by creating and subsequently referring to a WDO that documents the classes related to the project of interest, and start to create instances of the Data and Method classes contained in the WDO. As these classes are instantiated, the instances are represented graphically in the SAW with the notation previously described.

The instances that effectively document a scientific process in a SAW can be customized by assigning labels that can serve as differentiators between different instances of the same class. Additionally, formats can be assigned to instances of Data classes to ground their specific representation. For example, the instance `3D Velocity Model` in Figure 1 could be assigned the format `mime:Text`.

Lastly, as the instances are introduced to the SAW and their corresponding graphical representations are created, additional information about their graphical layout position is attached to the instances.

## 3.2 Proof Markup Language

The goal of capturing provenance about data is to support the explanation of how data is created or derived, e.g., which sources were used, who encoded the data, and more. The Proof Markup Language (PML) ontology defines primitive concepts and relations for representing provenance about data. PML is divided into three parts [7]:

- The *provenance ontology* (PML-P) defines concepts to represent identifiable things from the real world that are useful to determine data lineage. For example, sources such as organization, person, agent, service, and others are included in PML-P;
- The *justification ontology* (PML-J) defines concepts and relations to represent dependencies between identifiable things;
- The *trust relation ontology* (PML-T) defines concepts and relations to represent belief assertions and

to use those assertions in explanations about data. The use of PML-T is beyond the scope of this paper and consequently is not included in the discussion below.

The foundational concept in PML-P is `pmlp:IdentifiedThing`, which refers to an entity in the real world. These entities have attributes that are useful for provenance, such as name, description, create date-time, authors, and owner. PML includes two key subclasses of `pmlp:IdentifiedThing` motivated by provenance representational concerns: `pmlp:Information` and `pmlp:Source`. The concept `pmlp:Information` supports references to information at various levels of granularity and structure. The concept `pmlp:Source` refers to an information container, and it is often used to refer to all the information from the container. A `pmlp:Source` could be a document, an agent, a web page, among others. PML-P provides a simple but extensible taxonomy of sources.

PML-J provides concepts and relations used to encode the information manipulation steps used to derive a conclusion. A justification requires concepts for representing conclusions, conclusion antecedents, and the information manipulation steps used to transform/derive conclusions from antecedents. Although these terms stem from the theorem proving community they can be mapped to more familiar workflow terms; for example, conclusions refer to intermediate data and antecedents refer to the inputs of some processing step. The justification vocabulary has two main concepts: `pmlj:NodeSet` and `pmlj:InferenceStep`. A `pmlj:NodeSet` includes structure for representing a conclusion and a set of alternative `pmlj:InferenceSteps` each of which can provide an alternative justification for a conclusion. Lastly, every `pmlj:NodeSet` has a unique web-addressable identifier, i.e., has a URI. Web-addressable `pmlj:NodeSets` make it possible to construct justification trees in a distributed environment.

Figure 2 shows an example of provenance encoded in PML. The example is in reference to the part of the process shown in Figure 1 where a 3D Velocity Model is obtained from the execution of the Tomo method, which takes as input Ray Coverage, Shot Points, and a Gridded Time Field. Line 2 of Figure 2 starts the declaration of the `pmlj:NodeSet`, which is divided into two main parts: a conclusion (lines 3-12) and the inference step that led to the conclusion (lines 13-35). The conclusion is described as `pmlp:Information` (lines 4-11), an identifiable thing that has a specific URL and format. The inference step is described in the terms of the inference engine used to carry out the inference step (lines 15-17), the in-

```

1 <rdf:RDF>
2   <pmlj:NodeSet rdf:about="http://.../Tomo.owl#answer">
3     <pmlj:hasConclusion>
4       <pmlp:Information>
5         <pmlp:hasURL>
6           http://.../smoothed-3D-velocity-model.dat
7         </pmlp:hasURL>
8         <pmlp:hasFormat>
9           http://.../3D-model.owl#model
10        </pmlp:hasFormat>
11      </pmlp:Information>
12    </pmlj:hasConclusion>
13    <pmlj:isConsequentOf>
14      <pmlj:InferenceStep>
15        <pmlj:hasInferenceEngine>
16          http://...#holes
17        </pmlj:hasInferenceEngine>
18        <pmlj:hasInferenceRule>
19          http://...#tomo
20        </pmlj:hasInferenceRule>
21        <pmlj:hasAntecedentList>
22          <pmlj:NodeSetList>
23            <ds:first>
24              http://.../slowness-perturb.owl#answer
25            </ds:first>
26            <ds:next>
27              http://.../ray-coverage.owl#answer
28            </ds:next>
29            <ds:last>
30              http://.../ray-perturb.owl#answer
31            </ds:last>
32          </pmlj:NodeSetList>
33        </pmlj:hasAntecedentList>
34      </pmlj:InferenceStep>
35    </pmlj:isConsequentOf>
36  </pmlj:NodeSet>
37 </rdf:RDF>

```

Figure 2: PML NodeSet representing an execution of the holewdo:Tomo processing step

ference rule applied (lines 18-20), and the antecedents used to for the inference rule (lines 21-33) where each antecedent is encoded using the URI of its corresponding `pmlj:NodeSet`. Relating to workflow terms, the inference engine can be mapped to a workflow executing environment, the inference rule can be mapped to a specific activity executed within a workflow, and the antecedents can be mapped to the inputs taken by that activity.

## 4 Capturing Provenance

This section describes the systematic approach of enhancing a scientific process to capture provenance. Figure 3 shows an overall view of the approach and the subsequent sections describe the details.

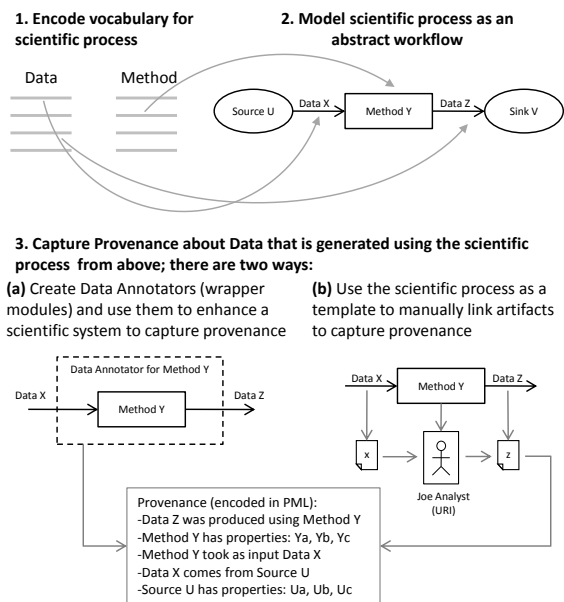


Figure 3: The process of capturing provenance with Abstract Workflows

## 4.1 Modeling the Scientific Process

As illustrated in Figure 3, documenting the scientific process is intended to be a scientist-driven process started by the identification of vocabulary terms. The vocabulary is encoded as an OWL ontology referred to as a Workflow-Driven Ontology (WDO) and as described in Section 3.1.1. The Data and Method terms of the scientific process are elicited by the scientist either by referencing terminology used in related systems, identifying terms through communal consensus, or referring to other well established vocabularies.

Based on the vocabulary captured in the WDO, the scientist continues to document a scientific process as a Semantic Abstract Workflow (SAW) as described in Section 3.1.2 and illustrated in Figure 3.

## 4.2 Capturing provenance for automated processes

Once a SAW has been authored to model a process from the perspective of the scientist, it can be used to drive the generation of “data annotators”. One data annotator module is created and tailored for each Method instance (or workflow activity) included in the SAW. Subsequently, the set of data annotator modules created from a SAW are used to instrument the system or executable workflow that carries out the scientific process to capture provenance. The instrumentation of automated processes is discussed in the next subsection.

Data annotators are modules designed to capture

provenance associated with automated workflow activities. Data annotators are built for the sole purpose of logging provenance rather than transforming data; therefore, data annotators are 1) non-intrusive from the perspective of the scientific processing, and 2) use a provenance language as the exclusive means of communication (i.e., input and outputs of data annotators are provenance elements). When using data annotators, provenance is transformed by each annotator by always enhancing the input provenance trace with more information. In our current efforts, we have used PML to encode provenance; therefore, the inputs and outputs of all data annotators are always `pml.j:NodeSets` as discussed in Section 3.2.

The logging mechanism of data annotators is divided in three phases:

- *Data Capture*: Provenance includes intermediate workflow results (i.e., workflow activity outputs); therefore, data annotators are responsible for capturing the outputs of the workflow activity it is tailored for.
- *Composition*: Once the output data has been captured or intercepted for a workflow activity, it is composed with the provenance related to the activity’s inputs.
- *Forward*: Once provenance has been constructed from the composition of provenance from the inputs of a workflow activity plus the provenance related to the workflow activity itself, it is available to be used by other data annotators that may be executed in the future. Because each annotator propagates its antecedents (i.e., its inputs) to the next annotator, there is a full justification available at any point in the execution of a workflow.

### 4.2.1 In-process vs. Post-process annotation

Once the data annotators are generated from the SAW, the system or executable workflow that carries out the scientific process needs to be instrumented to invoke the data annotators. This task requires the assistance of a technically-oriented user that is familiar with the system to be enhanced. This is because certain properties of a system will dictate when provenance should be logged. For example, when intermediate artifacts are not persisted during execution of the workflow, an in-process approach must be used to capture these intermediate artifacts before they are expunged from the process. This implies, however, that the workflow has to be modified to invoke data annotators at precise moments in execution, and thus, the coordinating agent of the data annotators is also the coordinating agent of the executable workflow. Deciding where to add the data annotator calls on a system requires knowledge about the inner-workings of the

system, such as what parts correspond to process coordination (i.e., process control) and which parts correspond to the execution of workflow activities.

If a workflow does not delete intermediate results or if users are unable to modify a workflow, then the non-invasive post-process annotation can be used. In this case, knowing about workflow how/when/where workflow activities are invoked is less important than knowing specific properties of data output from the activities. This is because, a post-processing annotators search for the existence of certain types of data with certain properties, which signifies that a particular workflow activity was executed. For example, if an annotator was configured to capture provenance associated with the Hole's activity "generate velocity model" it would search the file system for the existence of a "3D model", which would provide evidence that the "generate velocity model" was executed.

#### 4.2.2 Centralized vs. Distributed Provenance

The provenance captured by data annotators is encoded in PML, which can be constructed from distributed NodeSets; thus, data annotators can too be distributed along with any remote services that are invoked by a workflow. This is possible because the inputs to data annotators, which are PML NodeSets associated with executions of workflow activities, are referenced by URIs. This is convenient because often times complex scientific processes are modularized and controlled by a master script that in turn makes calls to services which may be located remotely. In these cases, the agent coordinating the data annotators does not need to know about provenance as a whole, but only encounters the URIs of intermediate provenance elements.

Computational platforms, including scientific workflow environments such as Kepler [6] and Taverna [5], have been extended to record provenance: meta data about services invoked by their workflow engines and about input and output information consumed and produced by these services. Specifications that are executed by workflow engines, however, tend to be unaware of services indirectly invoked during execution, i.e., other services called within services that are invoked by a workflow engine. In this context, we see that results of workflow executions would benefit of a distributed approach for capturing provenance consisting of aggregating provenance-related information from the individual services executed on behalf of the process to build provenance for the entire process.

### 4.3 Capturing provenance for manual processes

For the case of scientific processes that are carried out by humans, the approach to capture provenance is done manually by linking artifacts with respect to the SAW. The SAW that is used to model the scientific process serves as a template that is gradually filled with artifacts that result from human activity. For example, a manual scientific process may include a "Data analysis" activity that is carried out by a domain expert. The artifact resulting from the analysis activity, e.g., a document reporting the outcome of the analysis, is linked to the output of the corresponding activity in the SAW.

Once the scientific process is carried out to completion, at least one branch of the SAW template will be filled from start to finish. In other words, the SAW may contain several starting points that lead to the final outcome, but only one branch is needed to document the provenance of the final outcome. This is because while the goal of Semantic Abstract Workflows (SAWs) is to document scientific processes that are used to create scientific artifacts, the goal of PML is to document provenance about the creation of one scientific artifact. Consider Hole's Code use case; while the SAW in Figure 1 shows that the process to produce `holewdo:3D Velocity Model` is a loop consisting of five steps, provenance encoded in PML about the creation of a specific `holewdo:3D Velocity Model` would indicate the specific methods, as well as the number of iterations that were actually executed to create the end result.

## 5 Observations from Ongoing Efforts

In addition to the use case presented above, our approach to document and capture provenance about scientific processes is currently being used in several projects from the CyberShARE Center of Excellence ([www.cybershare.utep.edu](http://www.cybershare.utep.edu)), and the National Center for Atmospheric Research (NCAR) ([www.ncar.ucar.edu](http://www.ncar.ucar.edu)). CyberShARE alone encompasses the geoscience and environmental science domains providing a diverse set of use cases from which to evaluate our provenance logging techniques. Specifically, one of the environmental science projects is based on the execution of a workflow that has some activities which require human intervention. Reflectance data is captured in the Arctic by sensors attached to a cart that travels down a 300 meter tram. Upon completion of a run, the recorded data sets are transferred to a computer, by a human, from which processing of the data can resume. Additionally, there are steps in the workflow that require the use of software that requires human interaction. Based on the characteristics of the workflow, a post-processing data annotation

approach was used to piece together the justifications for data dumped from the workflow activities (i.e., constructing the PML-J links), which proved to be a success in terms of capturing all the steps the scientists were interested in. However, at the time of this work, there does not exist mechanisms for generating the associated identifiable things of the provenance (i.e., the PML-P artifacts) and it is required that the scientist manually encode this information.

Given that PML employs concepts from the context of theorem proving, environmental scientists found it difficult to complement the generated PML-J with the correct instances of PML-P. This was one of the main motivations for harvesting PML-P instances at the level of SAWs. By including PML-P information in the SAW, the the data annotators that are generated from the SAW are tailored to adorn the output NodeSet with the PML-P instances identified in the SAW whether or not the provenance is captured as in- or post-process.

With respect to geoscience, the process of Hole’s Code discussed here, as well as the process of using gravity, magnetic, and receiver function geophysical data to construct 2 1/2 Dimension Crustal Structure models of the Earth are other ongoing efforts. SAWs have been authored that capture these processes at a level that communicates well to experts in a wide area of domains and provenance has been generated for some datasets. Numerous iterations of specifying the processes with SAWs included colleagues from various disciplines such as seismology, computer science, and computational math to identify the adequate concept names that would promote understanding of the process being documented across a variety of disciplines.

With regards to NCAR, our techniques were also used to capture provenance associated with their Quicklook process [8]. The Quicklook workflow is a fully automated workflow that runs from start to completion without any human interactions. This workflow is actually a sub-workflow in a much larger and complicated workflow known as the CHIP pipeline that processes radio images of the sun. The smaller Quicklook process does not produce scientific quality images but is used to get a “quick look” of solar images that are not at a resolution high enough to be considered scientific. Many of the intermediate results in the Quicklook process are persisted after execution. Additionally, at the time of this work, we were unable to get a hold of the working script that coordinates this process; thus, we opted for a post-processing annotation of the data that worked well for both computer scientists and domain scientists. However, the larger process encompassing Quicklook has many workflow steps in which the intermediate results are expunged leaving this larger CHIP process as a perfect candidate to employ an in-processing capturing of provenance. As a whole,

the capturing of CHIP provenance requires a hybrid approach in which some of the provenance is captured in-process while the portions of the process associated with Quicklook are captured post-runtime; a scenario that we had not considered before this work.

## 6 Related Work

Kepler builds on workflow abstractions with a “provenance recorder” [1], a mechanism for collecting provenance within a workflow. Adding a provenance recorder to a workflow allows for the collection of workflow building steps and workflow evolution, a method of capturing important instances or versions of a workflow as it is being modified. The Kepler provenance framework is coupled with the Kepler workflow environment resulting in a autonomous system that can both execute workflows and record associated provenance by attaching provenance listeners to the underlying engine. The logging capabilities are not distributed in the sense that if a workflow method makes a call to a subworkflow, Kepler would not be aware of this and would not record it in its provenance trace. Additionally, the provenance captured by Kepler is not distributed in the sense that the provenance is managed centrally by the Kepler engine.

MyGrid, from the e-science initiative, tracks data and process provenance of workflow executions [12]. The type of provenance recorded by MyGrid for cyberinfrastructure applications is analogous to the kind of information that a scientist records in a notebook describing where, how and why experimental results were generated. From these recordings, scientists are able to operate in three basic modes: (i) debug, (ii) check validity, and (iii) update mode, which refer to situations when, a result is of low quality and the source of error must be identified, when a result is novel and must be verified for correctness, or when a workflow has been updated and its previous versions are requested.

One of the main features that separates our provenance solution from others is the ability to record truly distributed provenance. This is possible because our data annotators are loosely coupled with the workflow engine. The workflow engine need only make calls to the data annotators and pass them the URI string output from the previously called annotator without any concern about what or how the data annotator is doing.

## 7 Conclusion

This paper describes how abstract workflows are used to guide the generation of code capable of capturing scientific workflow provenance encoded in PML. The provenance capturing approach demonstrates three interest-

ing properties: (i) it relies on the flexibility of abstract workflows for comprehensively describing scientific processes that may not be possible to be described with the use of concrete (or executable) workflows; (ii) it relies on existing code, e.g., scripts and workflow specifications, to identify how process steps are connected instead of imposing the encoding of control flow in abstract workflows; and (iii) the approach is flexible to accommodate the complexities a multitude of scenarios for executing scientific processes including centralized and distributed environments, manual, automated, and hybrid modes of execution. As a proof of concept, actual scientific process provenance in multiple domains were captured and encoded in PML using the provenance capturing approach.

Finally, we recognize that the approach may have limitations with respect to scalability. For example, manually instrumenting workflows to capture provenance may be a daunting task in environments that adapt dynamically to environmental changes. Furthermore, the manual process of instrumenting systems may be error-prone. Understanding the properties of scientific processes and their impact on our provenance capture approach are open issues to be explored.

## 8 Acknowledgments

This work was supported in part by NSF grants HRD-0734825 and EAR-0225670. Also, we would like to acknowledge the feedback received from anonymous reviewers.

## 9 Availability

Additional information about the WDO-It! tool used to create Workflow-Driven Ontologies and Semantic Abstract Workflows and to create data annotators can be found at

<http://trust.utep.edu/wdo>

Also, the Probe-It! tool is available for visualizing provenance encoded in PML and can be found at

<http://trust.utep.edu/probeit>

## References

- [1] ALTINTAS, I., BARNEY, O., AND JAEGER-FRANK, E. Provenance collection support in the Kepler scientific workflow system. In *Provenance and Annotation of Data* (2006), pp. 118 – 132.
- [2] GATES, A. Q., PINHEIRO DA SILVA, P., SALAYANDIA, L., OCHOA, O., GANDARA, A., AND RIO, N. D. Use of abstraction to support geoscientists' understanding and production of scientific artifacts. In *Geoinformatics: Cyberinfrastructure for the Solid Earth Sciences*, G. Keller and C. Baru, Eds. Cambridge University Press, To appear.
- [3] GUARINO, N. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In *Proceedings of SCIE* (1997), pp. 139–170.
- [4] HOLE, J. Nonlinear High-Resolution Three-Dimensional Seismic Travel Time Tomography. *Journal of Geophysical Research* 97(B5) (1992).
- [5] HULL, D., WOLSTENCROFT, K., STEVENS, R., GOBLE, C., POCOCK, M. R., LI, P., AND OINN, T. Taverna: a Tool for Building and Running Workflows of Services. *Nucleic Acids Research* 34, Web Server issue (2006), W729–W732. doi:10.1093/nar/gkl320.
- [6] LUDAŠČER, B., AND ET AL. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience* (2005). Special Issue on Scientific Workflows.
- [7] MCGUINNESS, D., DING, L., PINHEIRO DA SILVA, P., AND CHANG, C. PML2: A Modular Explanation Interlingua. In *Proceedings of the AAI 2007 Workshop on Explanation-aware Computing* (Vancouver, British Columbia, Canada, July 22-23 2007).
- [8] MCGUINNESS, D. L., FOX, P., PINHEIRO DA SILVA, P., ZEDNIK, S., RIO, N. D., DING, L., WEST, P., AND CHANG, C. Annotating and embedding provenance in science data repositories to enable next generation science applications. In *American Geophysical Union, Fall Meeting (AGU2008), Eos Trans. AGU, 89(53), Fall Meet. Suppl., Abstract IN11C-1052* (2008).
- [9] PINHEIRO DA SILVA, P., MCGUINNESS, D. L., AND FIKES, R. A Proof Markup Language for Semantic Web Services. *Information Systems* 31, 4-5 (2006), 381–395.
- [10] SALAYANDIA, L., PINHEIRO DA SILVA, P., GATES, A. Q., AND SALCEDO, F. Workflow-Driven Ontologies: An Earth Sciences Case Study. In *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing* (Amsterdam, Netherlands, December 2006).
- [11] VIDALE, J. Finite-Difference Calculation of Travel Times in Three Dimensions. *Geophysics* 55(5) (1990).
- [12] ZHAO, J., WROE, C., GOBLE, C., AND QUAN, R. S., AND GREENWEED, M. Using Semantic Web Technologies for Representing E-science Provenance. In *Proceedings of the 3rd International Semantic Web Conference* (November 2004), pp. 92–106.