8-1-2009

On the Use of Abstract Workflows to Capture Scientific Process Provenance

Paulo Pinheiro da Silva
University of Texas at El Paso, paulo@utep.edu

Leonardo Salayandia
University of Texas at El Paso, leonardo@utep.edu

Nicholas Del Rio
University of Texas at El Paso, ndel2@miners.utep.edu

Ann Q. Gates
University of Texas at El Paso, agates@utep.edu

Comments:
Technical Report: UTEP-CS-09-24

On the Use of Abstract Workflows to Capture Scientific Process Provenance

Paulo Pinheiro da Silva, Leonardo Salayandia, Nicholas Del Rio, and Ann Q. Gates
Computer Science Department
University of Texas at El Paso, El Paso, Texas 79968
Email: see http://www.cs.utep.edu/paulo

***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 specification of an orchestration logic to execute a scientific process, and instead, are intended to document scientific processes as understood by scientists. Hence, abstract workflows can be specifically designed to capture the details of scientific processes that are relevant to the scientist with respect to provenance. In addition, abstract workflows are coupled with a representation of provenance that can accommodate distributed provenance-generating source code. We also 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.**

I. Motivation

Most scientists capture provenance about their data and process executions and record it in the form of logs, meta data, annotations, and others. However, it is often hard for other scientists to reuse provenance. For example, scientists may store collected provenance in databases that may not be designed to support sophisticated provenance-based queries in support of scientific activities such as understanding scientific data. A common representation for provenance, i.e., a provenance language, has been identified as a necessity to facilitate the reuse of provenance by scientists. 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 [1] as our preferred domain-independent language for encoding provenance.

Computational platforms including scientific workflow environments such as Kepler [2] and Taverna [3] 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.

The task of capturing provenance is a downside of the use of distributed provenance since it may be much harder to be accomplished in a distributed environment than in a centralized one. In this paper, we discuss the issue of enhancing scientific processes with automated ways of capturing provenance whether these processes are specified as scientific workflows or not, and whether they are centralized or distributed. The use of abstract workflows [4], which are not committed to be executed by machines, has been developed to facilitate the sharing of process knowledge (as defined in [5]) among scientists. Because of this flexibility, these specifications are neither restricted to boundaries of a cyber-environment nor to any specific technology for executing workflow specifications. This means that these abstract workflows are capable of describing how scientific processes can be executed in distributed environments. In this paper, we describe how abstract workflows have been used to generate code that enhances distributed scientific processes with functionalities for receiving, capturing and propagating provenance, namely the data provenance annotators. The paper also demonstrates that the use of data provenance annotators establishes a systematic approach for enhancing scientific processes with distributed provenance.

The rest of the paper is organized as follows. Section II introduces a scientific process use case for distributed provenance. Section III describes the languages and tools used for capturing distributed provenance for the scientific process use case. Section IV described the provenance capturing approach. Section V reports on other efforts to use the provenance capturing approach. Section VI describes approaches for capturing provenance used in other computational platforms. Conclusions are presented in Section VII.

II. Use Case - Hole's Code

To illustrate how our techniques can be integrated into complex workflows, we will refer to a process that cre-

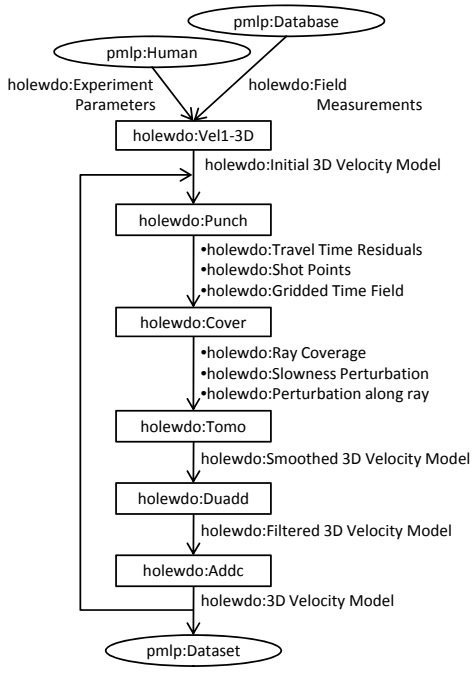

Fig. 1. Semantic Abstract Workflow for Hole's Code

ates a seismic velocity model of the Earth's crust using a nonlinear tomography inversion procedure [6] and a finite difference calculation [7] using 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 (`holewdo:Vel1-3D`). Next, the velocity model is refined gradually by executing an iterative process that uses seismic waves generated from controlled shot-point explosions (`holewdo:Punch`) and corresponding measurements of the arrival times of the waves at geophone stations (`holewdo:Cover`). The last three steps of the iteration correspond to a smoothing step (`holewdo:Tomo`), a filtering step (`holewdo:Duadd`), and a step to incorporate the refinements to the velocity model (`holewdo:Addc`). Such models are useful for earthquake analysis and oil exploration.

III. Background

A. 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 [4] 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) [8] , 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.

1) Workflow-Driven Ontologies: Guarino [5] 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 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 [8], the intention of WDOs is to allow scientists to capture process-related classes by extending the hierarchies of *Data* and *Method*. Furthermore, *Data* and *Method* classes can be related through *isInputTo* and *isOutputOf* relations to capture their data-flow interdependencies with respect to a scientific process.

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 vertices representing data flow between those activities. Activities connected through vertices 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) 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 `holewdo: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.

B. 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 [9]:

- The *justification ontology* (PML-J) defines concepts and relations to represent dependencies between identifiable things;
- 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 *trust relation ontology* (PML-T) defines concepts and relations to represent belief assertions and to use those assertions in explanations about data.

The goal of the justification ontology is to provide the 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. Figure 2 outlines a PML node set capturing the processing step implemented by `holewdo:Tomo` in Figure 1. The output of `holewdo:Tomo` is a `holewdo:Smoothed 3D Velocity Model` and this data is captured in the *Conclusion* element as an instance of `pmlp:Information`, described below. Additionally, the inputs consumed by `holewdo:Tomo` are captured as *Antecedents* of the node set's inference step. The term `pmlj:NodeSet` is chosen because it captures the notion of a set of nodes (with inference steps) from one or many proof trees deriving the same conclusion. Every `pmlj:NodeSet` has exactly one unique identifier that is web-addressable, i.e., a URI.

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. For example, in Figure 2 the node set is adorned with PML-P instances that effectively convey that this node set corresponds to an execution of `holewdo:Tomo`. The PML-P *inference engine* instance is named "Holes" to indicate that this captured step is part of a bigger process known as Holes's Code. Furthermore, the PML-P *inference rule* instance describes the specific step, (e.g., `holewdo:Tomo`) in terms of what the step does and what organization is responsible for this particular implementation of the algorithm. 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.

IV. Capturing Provenance

While the goal of 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

```
<rdf:RDF>
 <NodeSet rdf:about="http://.../Tomo.owl#answer">
  <hasConclusion>
   <pmlp:Information>
    <pmlp:hasURL rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://.../smoothed-3D-velocity-model.dat</pmlp:hasURL>
    <pmlp:hasFormat rdf:resource="http://.../registry/FMT/3D-model.owl#model"/>
   </pmlp:Information>
  </hasConclusion>
  <isConsequentOf>
   <InferenceStep>
     <hasInferenceEngine rdf:resource="http://.../pmlp/Holes.owl#holes"/>
    <hasInferenceRule rdf:resource="http://.../pmlp/Tomo.owl#Tomo"/>
    <hasAntecedentList>
     <NodeSetList>
      <ds:first rdf:resource="http://.../proof/slowness-perturb.owl#answer"/>
      <ds:next rdf:resource="http://.../proof/ray-coverage.owl#answer"/>
      <ds:last rdf:resource="http://.../proof/ray-perturb.owl#answer"/>
     </NodeSetList>
    </hasAntecedentList>
   </InferenceStep>
  </isConsequentOf>
 </NodeSet>
</rdf:RDF>
```

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

that were executed to create the end result. The following subsections show how SAWs are leveraged to systematically capture provenance about scientific artifacts.

A. Data Annotation

Once a SAW has been authored in WDO-It! it can be used to drive the generation of "data annotators" that are modules designed to capture provenance associated with workflow activities. Executing a set of data annotators corresponding to a single SAW is similar to executing a workflow in the sense that some coordinating agent is needed for both the synchronized invocation of each data annotator and for the message passing facilities needed for communicating between them.

Data annotators are built for the sole purpose of logging provenance rather than transforming data; therefore, data annotators use provenance as the exclusive language for 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 NodeSets.

The logging capability provided by data annotators can be decomposed into two 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, it is then composed with other provenance, such as information about the workflow activity being logged and encoded in PML.

The data capture phase must be implemented manually because there is no knowledge in SAWs regarding the specifics of where or how data will be generated. Therefore, data annotators by default are not fully executable and serve as skeletons that need to be enhanced by a user who has knowledge about the concrete workflow. In terms of PML, this means that data annotators are unable to capture all of the information contained in a PML conclusion by default. Although the SAW knows about the PML-P formats of the output information, it does not know where or when this data will reside and, thus, cannot automatically complete the "hasURL" element in the conclusion. On the other hand, the generation of software that implements the composition phase can be done automatically, and driven entirely from information captured in a SAW. For example, each SAW method knows about the format of the corresponding input/outputs. This knowledge is used to ensure that the data annotator correctly stores the output data in a PML conclusion as a reference, as its native representation, or as some XML friendly representation. For example, the node set in Figure 2 captures the output "smoothed 3D model" as a URL rather than stuffing the actual model into the XML. This is because the data annotator knows about the format "3D model", which can become quite large, thus opting for a pass-by-reference approach. It is important to note that without knowledge from the SAW about data formats, this automation would not be possible. Additionally, information about the inputs and any PML-P instances associated with a SAW method are also used for setting the NodeSet antecedents and injecting source information respectively.

In any case, an annotator must capture provenance of the workflow activity so that new PML NodeSets can be composed, where the conclusion is set to the captured output data and the antecedents reference the inputs that themselves are PML NodeSets associated with the preceding methods. Because each annotator propagates its antecedents (i.e., inputs) to the next annotator, at any point in the execution of a workflow, there is a full justification available.

1) In-Processing vs. Post-Processing Annotation: Certain properties of a workflow will dictate when provenance should be logged. For example, when intermediate artifacts are not persisted during execution of the workflow, an in-processing approach must be used to capture these intermediate artifacts before they are expunged from the process. This implies, however, that the workflow be modified to invoke data annotators at precise moments in execution, thus the coordinating agent of the data annotators is also the coordinating agent of the concrete workflow. Deciding where to add these calls to a workflow requires that a user understand specifics of a concrete workflow, such as what parts correspond to the coordination of process (i.e., control flow) and which parts correspond to the execution of workflow activities; for it is this knowledge that is needed to instrument the workflow.

If a workflow does not delete intermediate results or if users are unable to modify a workflow, then the non-invasive post-processing 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.

2) Centralized vs. Distributed Provenance: The provenance captured by data annotators is encoded in PML, which is distributed; 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 the dependent 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 or may not be located remotely, as is the case with Hole's code. 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.

B. PML-P Harvesting and Generation

Instances of PML-P concepts such as the inference rule `Tomo.owl#Tomo` in Figure 2 mentioned are often annotated by other instances of PMP-P concepts. For example, the PML-P of the inference rule above may identify the creators of the rule by having a reference to an instance of `pmlp:Person` as the value for its `hasAuthor` property. As we can observe from the example above, instances of PML-P concepts are used to annotate both other instances of PML-P concepts and instances of PML-J concepts. Moreover, in an ideal situation, PML-P concepts are intended to be reused.

In the context of data provenance annotators, PML-P instances should first be harvested to prevent the creation of multiple PML-P instances about a single real-life object. In the context of scientific workflows, tools like WDO-It! should be able to harvest instances of PML-P concepts that map into SAW concepts. For example, a `wdo:Method` instance can be used to describe a tool functionality and it is known that the `wdo:Method` concept maps into the `pmlp:InferenceRule concept`. Thus, once a `wdo:Method` is instantiated during the creation of a SAW, WDO-It! may harvest PML-P concepts by allowing users to search and select instances of `pml-p:InferenceRule` to describe the new instance of a `wdo:Method`.

V. 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), and the National Center for Atmospheric Research (NCAR) (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 again requires human interaction leaving the concrete workflow fragmented in terms of execution. Based on the characteristics of the workflow, a post-processing data annotation approach was used to piece together the PML-J from data dumped from the workflow activities, 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 PML-P artifacts and required that the scientists manually encode this information. The PML-P ontology is loaded with many concepts and these concepts were defined in the context of theorem proving; therefore, it was difficult for environmental scientists to annotate the captured 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. This allows that the data annotators to come already configured to adorn the output node sets with the PML-P instances identified in the SAW whether or not the provenance is captured via pre or post processing.

The geological processes associated with Hole's Code and 2 $1/2$ Dimension Crustal Structure of the Earth are more ongoing efforts. SAWs have been authored that capture the process 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 other 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 [10]. 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.

VI. Related Work

Kepler builds on workflow abstractions with a "provenance recorder" [11], 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 cyber-infrastructure 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 identied, when a result is novel and must be veried for correctness, or when a workow 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.

VII. 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 two interesting 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 both centralized and distributed environments. As a proof of concept, actual scientific process provenance in multiple domains were captured and encoded in PML using the provenance capturing approach.

Acknowledgment

This work was supported in part by NSF grants HRD-0734825 and EAR-0225670.

References

[1] P. Pinheiro da Silva, D. L. McGuinness, and R. Fikes, "A Proof Markup Language for Semantic Web Services," *Information Systems*, vol. 31, no. 4-5, pp. 381–395, 2006.

[2] B. Ludaścher and et al., "Scientific Workflow Management and the Kepler System," *Concurrency and Computation: Practice & Experience*, 2005, special Issue on Scientific Workflows.

[3] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a Tool for Building and Running Workflows of Services," *Nucleic Acids Research*, vol. 34, no. Web Server issue, pp. W729–W732, 2006, doi:10.1093/nar/gkl320.

[4] A. Q. Gates, P. Pinheiro da Silva, L. Salayandia, O. Ochoa, A. Gandara, and N. D. Rio, "Use of abstraction to support geoscientistsúnderstanding and production of scientific artifacts," in *Geoinformatics: Cyberinfrastructure for the Solid Earth Sciences*, G. Keller and C. Baru, Eds. Cambridge University Press, To appear.

[5] N. Guarino, "Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration," in *Proceedings of SCIE*, 1997, pp. 139–170.

[6] J. Hole, "Nonlinear High-Resolution Three-Dimensional Seismic Travel Time Tomography," *Journal of Geophysical Research*, vol. 97(B5), 1992.

[7] J. Vidale, "Finite-Difference Calculation of Travel Times in Three Dimensions," *Geophysics*, vol. 55(5), 1990.

[8] L. Salayandia, P. Pinheiro da Silva, A. Q. Gates, and F. Salcedo, "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.

[9] D. McGuinness, L. Ding, P. Pinheiro da Silva, and C. Chang, "PML2: A Modular Explanation Interlingua," in *Proceedings of the AAAI 2007 Workshop on Explanation-aware Computing*, Vancouver, British Columbia, Canada, July 22-23 2007.

[10] D. L. McGuinness, P. Fox, P. Pinheiro da Silva, S. Zednik, N. D. Rio, L. Ding, P. West, and C. Chang, "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.

[11] I. Altintas, O. Barney, and E. Jaeger-Frank, "Provenance collection support in the Kepler scientific workflow system," in *Provenance and Annotation of Data*, 2006, pp. 118 – 132.

[12] J. Zhao, C. Wroe, C. Goble, R. S. andq D. Quan, and M. Greenweed, "Using Semantic Web Technologies for Representing E-science Provenance," in *Proceedings of the 3rd International Semantic Web Conference*, November 2004, pp. 92–106.