9-1-2007

WDO-It! A Tool for Building Scientific Workflows from Ontologies

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

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

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

Comments:
Technical Report: UTEP-CS-07-50

WDO-It! A Tool for Building Scientific Workflows from Ontologies

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

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

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

Abstract

One of the factors that limits scientists from fully adopting e-Science technologies and infrastructure to advance their work is the technical knowledge needed to specify and execute scientific workflows. In this paper we introduce WDO-It!, a scientist-centered tool that facilitates the scientist's task of encoding discipline knowledge in the form of workflow-driven ontologies (WDOs) and presenting process knowledge in the form of model-based workflows (MBWs). The goal of WDO-It! is to facilitate the adoption of e-Science technologies and infrastructures by allowing scientist to encode their discipline knowledge and process knowledge with minimal assistance from technologists. MBWs have demonstrated potential to facilitate knowledge understanding and transfer of key scientific processes for a wide range of people, including students. An experimental version of WDO-It! has been developed as a proof of concept and scientists are using the tool to develop experimental WDOs and MBWs for several scientific disciplines.

1 Introduction

Ontologies that encode knowledge about a particular discipline and that are accessible through distributed environments such as the Web have the potential to change the way in which research is conducted. Indeed, this technology and other e-Science technologies are providing scientists with the ability to discover, access, integrate and disseminate knowledge.

A challenge in fulfilling the promise of ontologies is to leverage the encoded knowledge to support technology that facilitates the design and execution of scientific workflows that compute results through the composition of software services. Productive design of scientific workflows often depends on the effectiveness of the communication between the discipline domain experts and technology experts, including their ability to share their specific needs in the design of the workflow. Indeed, the development of workflows that can be executed on a computer require a significant amount of detail to specify, which results in high interaction between the scientist and technology expert. A solution is to support the specification of workflows at different levels of abstraction, depending on the needs of the user. These abstract workflows can capture the steps or processes for generating a result of a given type without detail, e.g., details concerning interoperability, orchestration, and the binding of abstract methods to services. While such notions are important for workflow execution, discipline experts should not have to be concerned with execution details, but rather on understanding and sharing the process.

This paper presents WDO-It!, a novel tool for defining ontologies that enable domain experts to encode concepts categorized as data (e.g., raw measurements and derived data such as maps and graphs) and methods (i.e, services, processes, or functions) and relationships among these concepts. This type of ontology, which we call a Workflow-Driven Ontology (WDO) [13], extends a traditional ontology. More importantly, it is one that supports composition of software applications that conciliate the needs of domain and computer scientists. WDO-It! allows scientists to develop a WDO, generate abstract workflows from the WDO, and refine the WDO based on evaluation of the abstract workflow. WDO-It! abstract workflows are called Model-Based Workflows (MBWs) [12] and are scientist-centered. MBWs can be input into a workflow management system for workflow instantiation and execution. WDO-It! was developed to address a real need of scientists who are developing the Gravity Ontology [3] who needed to record a collection of data-method relationships, e.g., “isOutputOf”, “getsInputFrom”, and to navigate through these relationships in ways that would not be naturally supported by traditional ontology editors like Protege [10] and Swoop. WDO navigation requirements are indeed the steps for traversing ontologies both to generate and to render abstract workflows as later described in Section 4. WDO-It! was developed to replace a partial encoding of the Gravity Ontology that was developed using MS Excel due to the complexity of maintaining it through traditional ontology editors.

In addition to Section 4, this paper includes the following sections. Section 2 introduces the WDO-It! tool and shows how it is used to encode domain knowledge and to use this domain knowledge to capture process knowledge. Section 3 describes the key WDO concepts and how these concepts are organized in the WDO meta-model. Section 5 describes our efforts to assess and improve WDO-It! Section 6 discusses related work including the comparison of WDO-It! to tools supporting the development of ontologies and workflows. Section 7 summarizes the main contribution of WDO-It!

2 WDO-It! In Use

2.1 Overview

If knowledge is going to be shared, it must be encoded in a way that agents other than the authors of the encoding can access, understand and reuse. In a knowledge sharing context, agents, whether authors or consumers of knowledge, are assumed to be humans or intelligent software systems. According to Guarino [4], ontologies can be categorized according to their level of dependence on a particular task or point of view. For example, *top-level ontologies* refer to ontologies that describe very general concepts like space, time, matter, etc. *Domain ontologies* and *task ontologies* specialize the terms used in top-level ontologies, where “domain” ontologies describe the vocabulary of a particular domain, e.g., seismology, and “task” ontologies describe the tasks or activities of a particular domain, e.g., using gravity data to create contour maps.

This paper is focused on how scientists as knowledge authors can use WDO-It! for encoding domain knowledge about their field and for using this encoding to capture and encode task knowledge in the form of abstract scientific workflows describing processes in their field. WDO-It! is a scientist-centered tool for developing and refining workflow-driven ontologies (WDOs), and that generates abstract workflow specifications. By using WDO-It!, scientists can use ontology concepts and relationships to create abstract workflow specifications. Further, WDO-It! enables scientists to better understand ontology concepts and relationships by allowing them to inspect how these elements are used to support abstract workflow specifications. For instance, by deciding whether abstract workflows generated from WDOs correspond to typical processes, scientists may change the workflows by changing the relationships between concepts that are specified in the ontology. In this sense, WDO-It! enables scientists to improve ontologies and workflow specifications incrementally. While existing tools such as Protege and Swoop provide solid solutions for general-purpose ontology development, these tools do not provide assistance towards specialized ontology development. As a result, general-purpose ontology development tools are difficult to use by non-expert users towards the development of specialized ontologies. The WDO-It! tool is geared towards the development of task ontologies, and it is being designed to assist the domain expert and not necessarily an ontology developer (or ontologist) to create task ontologies about their domains.

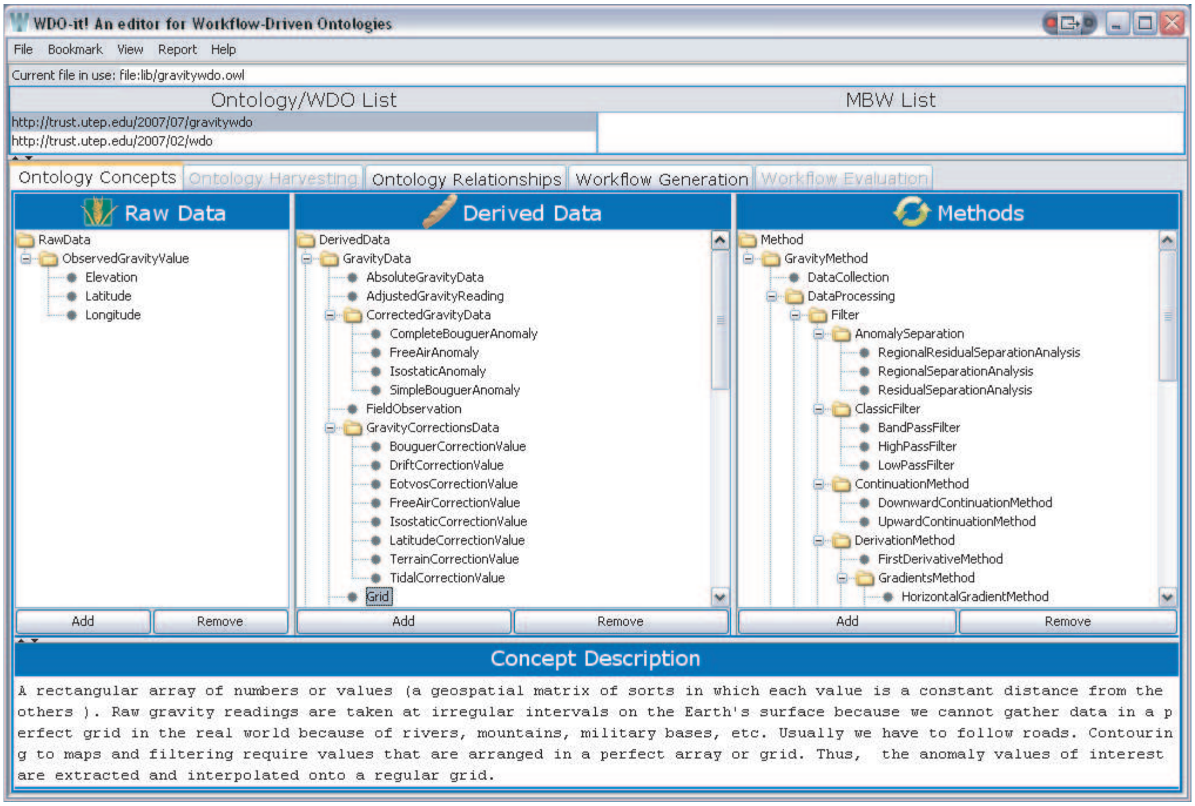
We anticipate that knowledge encodings generated through WDO-It! can and should be used for purposes other than the generation of scientific workflows, e.g., to support semantic search or information integration, although a discussion about these other purposes are beyond the scope of this paper.

2.2 Domain Knowledge Encoding

When it comes to encoding domain knowledge, scientists should be able to name and describe concepts related to real objects in their fields, e.g., sensors, datasets, tools. Further, scientists should be able to identify relationships between concepts by explaining how these objects are related, e.g., when explaining the relationship between a thermometer and a temperature dataset, a scientist should be able to state that the temperature dataset is an output of the thermometer. In terms of domain knowledge encoding, WDO-It! supports two functionalities: brainstorming and harvesting.

The brainstorming functionalities, provided through the *Concept* and the *Relationship* tabs in WDO-It!, support scientists in encoding concepts and concept relationships. Figure 1 presents a snapshot of the WDO-It! Concept Tab for the Gravity WDO that was derived from the Gravity Ontology [3]. In this area of the tool, scientists can identify concept names from their area of interest and classify them in three basic hierarchies: Raw Data, Derived Data, and Methods. The Raw Data concept hierarchy is intended to capture the concept elements that represent data in its simplest form, as viewed in the specific domain, e.g., field observation data. The Derived Data hierarchy is intended to capture concept elements that represent data that has been processed or modified by some application, and that the scientist community sees as valuable, e.g., filtered dataset, or a map artifact. Finally, the Methods hierarchy represents the concepts related to tool functionalities, services, algorithms or anything that is used to access and transform data. Concept specialization is specified at the time new concepts are added to the ontology. Under the *Relationship* tab, the scientist can identify and encode the relationships among these concepts.

The harvesting functionality of the tool, provided through the *Ontology Harvesting* tab, is used by scientists to encode concepts and relationship by referring to concepts from other ontologies and that are to be reused within the context of a workflow-driven ontology. The harvesting functionality is restricted to OWL ontologies [9] consider-



A rectangular array of numbers or values (a geospatial matrix of sorts in which each value is a constant distance from the others). Raw gravity readings are taken at irregular intervals on the Earth's surface because we cannot gather data in a perfect grid in the real world because of rivers, mountains, military bases, etc. Usually we have to follow roads. Contouring to maps and filtering require values that are arranged in a perfect array or grid. Thus, the anomaly values of interest are extracted and interpolated onto a regular grid.

Figure 1. Snapshot of WDO-It! showing the Concept Tab.

ing the larger availability of these kinds of ontologies.

WDO-It! uses a *Workspace* structure in support of the harvesting functionality. The workspace is composed of a list of ontologies and a list of abstract workflows called “MBW List,” and it is located above the Concept Tab as presented in Figure 1. Note in the figure that the abstract workflow list is empty, and the ontology list has two elements. The ontology on the top of the list represents the ontology under development and it is called the *base ontology*. In Figure 1, the base ontology is named gravitywdo (fully referred in the figure as http://trust.utep.edu/2007/07/gravitywdo) and the second ontology is the upper-level WDO ontology further described in Section 3.1. In the case of ontology harvesting, a base WDO ontology is created in the workspace, and one or more ontologies, either WDOs or other types of ontologies, are loaded into the workspace to be used as sources of concepts and relationships for the base WDO ontology. Ontologies can be imported at any time in the knowledge encoding process, and the Harvesting Tab enables users to add WDO concepts to the base ontology by referring to the imported ontologies.

2.3 Process Knowledge Encoding

At any time during the process of encoding knowledge, scientists may want to use existing concepts and relationships to generate an MBW that produces derived data. MBWs are said to be abstract workflows because they are not necessarily matched to any specific service implementation. Workflow functionalities, provided by the *Workflow Generator* and *Workflow Evaluation* tabs, are used by scientists to develop, visualize, and assess the quality of MBWs, and eventually to instantiate concrete workflows from MBWs, i.e., workflows based on existing services that can be executed through workflow engines.

Through the WDO-It! “Workflow Generation” tab, scientists are indeed asking the tool for suggestions on how to generate a given product from the data and methods in the WDO. In Figure 2, for example, the user is asking for an MBW that produces a contour map about gravity data for a given footprint. In the figure, `ContourMap` is selected from the derived data hierarchy on the left side of the figure. This hierarchy is the same one presented in Figure 1. By selecting a derived data concept, WDO-It! executes the workflow generation algorithm described in Section 4 to create the MBW shown on the right side of the figure. The scientist can then assess whether the MBW represents

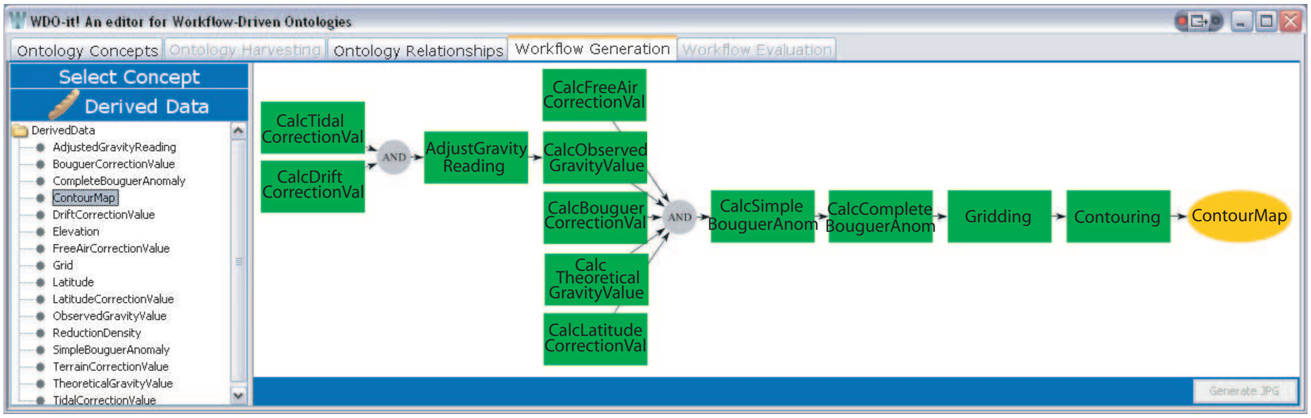


Figure 2. Snapshot of WDO-It! showing the Workflow Generation Tab.

a well-established process used in geophysics to produce contour maps from gravity data, or if it contains service compositions that are not accepted or used in the field. By endorsing MBWs, scientists are capturing process knowledge that complements the domain knowledge previously acquired by the knowledge encoding functionalities. By assessing and correcting MBWs, scientists can improve the encoded knowledge considering that MBW corrections are applied to concepts and relationships in the WDO used to generate the MBW. The "Evaluation" tab is the place were scientists can either endorse or correct MBWs.

3 Workflow-Driven Ontologies (WDO)

3.1 WDO Meta-model

The WDO meta-model is presented in Figure 3. The meta-model is rooted in the OWL concept *Thing*. The class *WDOConcept* represents the most general WDO class, and the model is then divided into two main hierarchies; the hierarchy rooted in *WFSequenceElement* contains subclasses that are used to specify workflow actions and control-flow; and the hierarchy rooted in *Data* that contains subclasses used to represent primitive data concepts of a scientific domain of interest, as well as classes used to compose complex data constructs that are both consumed by and derived from workflow actions. These complex data constructs are referred to as *Composite Data*, and are further discussed in the next section.

The *Method* metaclass is used to represent actions in a scientific domain of interest, e.g., services, algorithms, or application functionalities. Methods can be viewed as a special kind of workflow responsible for executing a single action (as presented in Figure 3, *Method* is a subclass of *WF* metaclass). Complex workflows can be composed of simpler workflows including methods. The input of these methods either comes from an agent, e.g., a human user, or from the output of a previously executed workflow according to the specified ordering in a more complex workflow specification. The end results of complex workflows are the outputs of the method that executes last. For example, a "sequential workflow" represented by the *WFSequence* metaclass is a composition of two workflows that can be, for example, two methods. In the case of a sequential workflow, the composition is supported by the *hasSuccessor* and *hasPredecessor* relationships connecting *WFSequence* to *WF* and *WFSequenceElement* respectively. The *WFSet* is the other WDO metaclass that supports the composition of workflows from other workflows. Each *WFSet* contains two or more workflows and its semantic indicates that the contained workflows may be run in parallel in the control-flow ordering. Notice that *WFSet* is said to be a

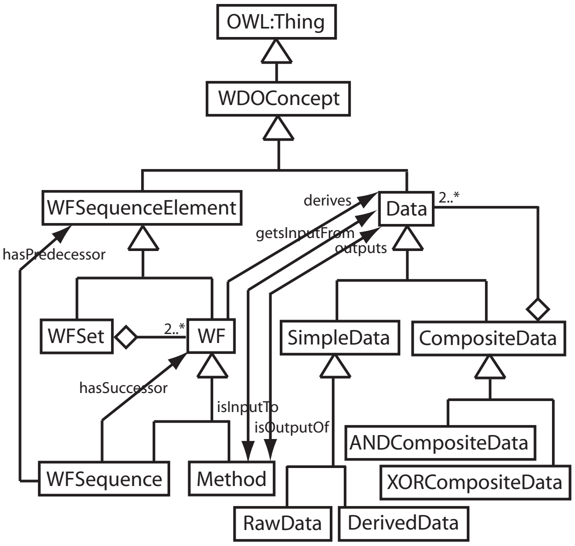


Figure 3. The WDO Metamodel.

set because it does not specify the exact ordering between the contained workflows. Ordering underspecification is a design goal of the WDO meta-model, as the resulting workflow specifications are considered abstract.

3.2 Composite Data and Relationships

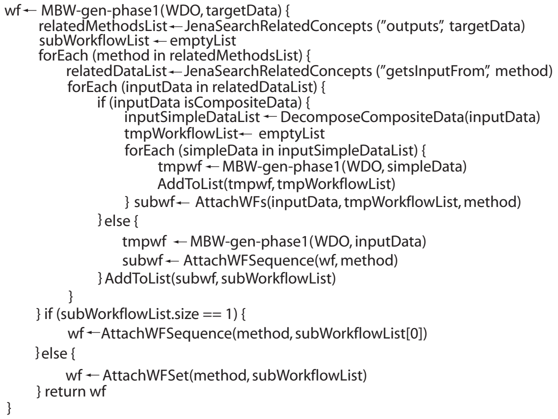
As presented in Figure 3, the *Data* meta-class is specialized into *SimpleData* and *CompositeData*. *SimpleData* is further specialized into *RawData* and *DerivedData* that are used to represent concepts that scientists identify in their fields. *CompositeData* is introduced in support of the encoding of process knowledge. *CompositeData* is further specialized into *ANDCompositeData* and *XORCompositeData* and it is used to construct complex data that can be related as input or output of a method. For example, the *ANDCompositeData* class can be used to specify that a method simultaneously requires two or more data input, whether these input data are of type *SimpleData* or other *CompositeData*. The *XORCompositeData* meta-class is used to specify alternative inputs or outputs of a method, meaning that a method requires exactly one component of the XOR composite data construct as input or that the method has different types of output. At the moment, WDO-It! allows scientists to specify multiple inputs to a method, which are automatically encoded as an *ANDCompositeData*, and alternative outputs for the method, which are automatically encoded as an *XORCompositeData*. The meta-model, however, has a provision for more complex *CompositeData* that could include combinations of *ANDCompositeData* and *XORCompositeData*.

4 Model-Based Workflows (MBW)

The Model-Based Workflow generation algorithm uses WDO concepts and relationships as input. The algorithm requires a user to identify a *target data* concept, which should be a Derived Data. In fact, Raw Data is intended to represent data that are not affected by methods, e.g., user input.

Once a target data is defined, the algorithm searches for WDO methods that specify the target data as their output. If more than one method is found in this step, a *WFSet* construct is used to specify a parallel ordering, indicating that there may be several branches in the workflow specification. The generation of each individual branch is then continued by searching for the input data required by the method identified for each branch.

The method may require either simple data or composite data as input. If the method requires composite data input, then the associated composite data is recursively broken down to its individual simple data components. Simple data components that are Raw Data are considered to be initial input to the workflow and the algorithm terminates for that workflow branch at that point. In terms of a complex workflow, a *terminal input* is data that is not produced by any method in the collection of WDOs used to generate an MBW. Raw data, by definition, are terminal inputs. Subsequently, the resulting workflows generated for each of the simple data components is attached to the particular branch of the workflow in the following way: if the number of simple data concepts identified as the input of the method of the branch is exactly one, then a sequential workflow construct is used to attach the current method concept to the generated subworkflow. If the number of simple data concepts identified is more than one, then a WFSet construct is used to specify the parallel specification dictated by the AND Composite Data or XOR Composite Data constructs previously decomposed. Finally, the workflow terminates for that branch if one of the following happens: there are no data concepts identified as input to the current method concept; or if there are no methods that derive the derived data used as input for the current method concept. For the second case, the derived data is assumed to be a terminal input.



```
wf ← MBW-gen-phase1(WDO, targetData) {
    relatedMethodsList ← JenaSearchRelatedConcepts (”outputs”, targetData)
    subWorkflowList ← emptyList
    forEach (method in relatedMethodsList) {
        relatedDataList ← JenaSearchRelatedConcepts (”getsInputFrom”, method)
        forEach (inputData in relatedDataList) {
            if (inputData isCompositeData) {
                inputSimpleDataList ← DecomposeCompositeData(inputData)
                tmpWorkflowList ← emptyList
                forEach (simpleData in inputSimpleDataList) {
                    tmpwf ← MBW-gen-phase1(WDO, simpleData)
                    AddToList(tmpwf, tmpWorkflowList)
                } subwf ← AttachWFs(inputData, tmpWorkflowList, method)
            } else {
                tmpwf ← MBW-gen-phase1(WDO, inputData)
                subwf ← AttachWFSequence(wf, method)
            } AddToList(subwf, subWorkflowList)
        }
    } if (subWorkflowList.size == 1) {
        wf ← AttachWFSequence(method, subWorkflowList[0])
    } else {
        wf ← AttachWFSet(method, subWorkflowList)
    } return wf
}
```

Figure 4. MBW Generator Pseudo-code.

On a second phase of the algorithm, the workflow is traversed from terminal inputs towards the target data concept. In this phase, the intention is to find additional branches specified by composite data related as output of method concepts. The alternate workflow branches identified here may or may not derive the target data concept, but they are important to specify in the final workflow specification so that the user is aware of alternate branches that may prevent the workflow from deriving the target data, e.g., when the alternate branches are executed instead of the intended main branch.

The graphical representation of generated Model-Based Workflows utilizes the following notation: Data concepts are represented by ovals, Method concepts are represented by rectangles, boolean connectors used to specify composite data are represented by appropriately labeled circles, and connecting arrows indicate the data flow specified by the workflow specification.

5 Evaluation Efforts

The best way of assessing the WDO-It! tool is through its actual use to encode ontologies and abstract workflows that scientists can sketch in a piece of paper. In terms of demographics, we have had seven technologists and four subject matter experts (SMEs), i.e., scientists, in three fields taking participation in these evaluations. The comprehensive evaluations described in this papers are long processes that require numerous face-to-face meetings between technologists and SMEs to be completed. The strategy used to evaluate WDO-It! consists of four tasks that are associated with an evaluation effort:

1. Establish training workshops for scientists that are centered on understanding the notion of workflows and services and how they can be used to encode knowledge and tasks that are useful to scientists.

2. Perform usability studies with scientists to determine the understandability and clarity of the notation used to represent abstract workflows.

3. Use WDO-It! to encode knowledge from different domains in OWL.

4. Generate MBWs out of WDOs and get the scientists to endorse these WDOs.

Three major WDOs and several minor WDOs have been developed so far. Major WDOs should describe concepts and relationships required for building a complex workflow in a given domain. Complex workflows are non-trivial compositions of services (normally ten or more services) like the one presented in Figure 2. Also, major WDOs should reflect a level of commitment by their developers by showing that most of the concepts have a participation in at least one relationship and that the key concepts are documented like the `grid` concept in Figure 1. The major WDOs developed so far are the following: the Gravity-WDO derived from the Gravity Ontology and used in the NSF GEON project; the Seismic-WDO derived from efforts of the NSF Earth Scope project; and the ARMA-WDO for radar signal processing.

The Gravity-WDO was the first major WDO to be created. This WDO was developed by scientists working with developers. The Gravity-WDO has demonstrated the need for both documenting encoded concepts and for generating reports as a single artifact. The Seismic-WDO was encoded by a WDO developer and the output was assessed by a scientist. Inheritance of relationships through data and method hierarchies is one of the major issues introduced by the Seismic-WDO. In this case, it is important to note that inherited relationships are critical for the generation of MBWs and that without relationship inheritance several MBWs could not have been generated. The ARMA-WDO was the first attempt to fully follow the evaluation strategy. MBW branching is an issue that was introduced by the ARMA-WDO ontology. Similar to relationship inheritance, the lack of support for branching prevents the generation of many MBWs. It is important to notice that help from a technologist was required to bootstrap the knowledge encoding process.

6 Related Work

The WDO-It! tool supports definition of workflow-driven ontologies and the generation of model-based workflows. To contrast the current capabilities with the WDO-It! tool, we examine related work in this section. The survey divides the landscape into two areas: ontologies that capture task-related concepts and the tools that specifically support them; and 2) workflow environments.

6.1 Current Capabilities: Ontology

There are several projects that include ontologies that describe task-related concepts. The Transparent Access to Biological Information Sources (TAMBIS) system [11]. TAMBIS is a bioinformatics source integration system that includes an ontology created with description logics independent of individual information sources. The ontology is used by scientists as a mediated schema to construct queries with concepts that are familiar to their areas of expertise. Queries formulated by the scientists are based on TAMBIS ontology concepts and TAMBIS transforms the queries into query plans that can be executed over individual sources. When first developed, TAMBIS was a novel approach because it introduced the use of an ontology as a global-schema that provide scientists with a conceptual-level view of diverse and heterogeneous information sources; however, the ontology supported by the TAMBIS system is fairly static, and scientists have little or no control on how TAMBIS creates query plans.

The myGrid project[1] aims to utilize information grid technology to provide middleware for data and application resource integration for the field of bioinformatics. The myGrid project includes a domain ontology that describes the bioinformatics research domain, and a service ontology that describes grid resources [14]. The classification provided by the first ontology is useful for finding or substituting services within a workflow. The second ontology is similar in

[1] http://www.mygrid.org.uk

nature to the OWL-S ontology described below, and supports the marshalling and configuring of web services for workflow implementation. Contributors of myGrid project are required to register grid services by providing descriptions of their services that are aligned with the myGrid ontology. The myGrid project targets a specific community (i.e., Bioinformatics), and as a result, they emphasize workflow reuse by providing a library of well established resources and workflows.

The Gene Ontology (GO) Consortium [1] is an organization that is developing a controlled vocabulary about gene information and providing a rich array of support tools. GO is split up into three main categories: the cellular component ontology, molecular function ontology, and the biological process ontology. GO supports an annotation mechanism that allows scientists to justify ontology contributions, as well as a set of tools to browse and maintain the ontologies and corresponding annotations. The GO Consortium emphasizes the need for a scientific organization to maintain and regulate ontology creation efforts, indicating that just information technology by itself will never be an ultimate solution for ontology development. Even though the GO contains terms and relations that can support the creation of biology workflow applications, the GO Consortium is not currently providing tools to provide this capability.

6.2 Current Capabilities: Workflow Environments

Scientific workflows are used to specify the composition of resources to achieve a complex scientific endeavor. There are many workflow environments that support specification and execution engines. With the Semantic Web as its basis, OWL-S [8] is a web service ontology that is based on the Ontology Web Language (OWL). OWL-S provides a service-composition specification language that allows workflows to be implemented from composition of web services. OWL-S is composed of three different parts: the service profile that provides additional information about the services, such as functionality, inputs, and outputs; the process model that provides information about how services are composed into a workflow; and the grounding that presents details about how to access the service. OWL-S provides a standardized language for general-purpose, service-composition specification that can be applied in any domain. OWL-S is geared toward use by software agents and not necessarily scientists.

There are environments that are specifically tailored to support scientists. Kepler [7], myGrids Taverna [5], and Pegasus [2] are workflow solutions specifically targeted for scientific use. The Kepler Scientific Workflow system allows users to construct scientific workflows using graphical notation and a GUI to visually program the workflow specification. Kepler generates and subsequently executes scientific workflows expressed in the Modeling Markup Language (MoML). MoML is a simple markup language that allows the specification of workflows that include actors and a director. Each actor carries on the execution of a step in the workflow, and the director gives the semantics of the control flow. Resources that are used as actors in the workflow specification require registration with the Kepler system, and the user requires training in workflow programming in order to be able to produce workflow specifications.

Taverna [5] is the workflow component of the myGrid project and is similar in nature to Kepler in that it provides a graphical notation for viewing program workflows. Taverna generates and executes Simple Conceptual Unified Flow Language (Scufl), a workflow language native to the tool. Taverna is designed specifically for the field of Bioinformatics, targeting a specific pool of resources. Taverna emphasizes workflow reusability by providing an extensive workflow library that users can access and customize to their specific needs. In scientific areas where diverse resources are not under the control of a central project, the tool may not be the best solution.

Pegasus [2] is a framework that maps complex scientific workflows onto distributed environments such as the Grid. Pegasus provides functionality to allow users to develop applications independently of a target execution system, and binding the abstract specification to a target system at a later time. Heuristic rules are used by Pegasus to bind the tasks specified by the “abstract workflow” to the resources that minimize the overall workflow execution time; it uses virtual data and provenance to generate and reduce the workflows based on data products that have already been computed. Pegasus provides great functionality for scheduling and resource reuse; however, the abstract workflows that Pegasus handles are templates of workflow specifications that are near to the technical workflow specifications that may not necessarily be scientific-user friendly. Pegasus relies on other tools, such as the Composition Analysis Tool (CAT) [6] to aid the scientist in producing an “abstract workflow.” These tools utilize existing knowledge-bases to aid the scientist, and the tools are not designed to change the knowledge bases during the process of generating the workflow. At a first glance, WDO-It! and CAT are similar in the sense that both systems make use of ontologies to create workflows and they both aim to enable non-workflow experts to build abstract workflows. There are, however, a number of differences between the two systems. In CAT, knowledge bases are created or probably (re)used with the primary purpose of supporting the generation of a given workflow that is to be later instantiated into an executable workflow. In WDO-It!, the final purpose is to enable scientists to create ontologies that describe their domain in a way that the ontologies can also be used to generate a collection of workflows in that domain.

7 Summary

This paper describes the WDO-It! tool that facilitates the scientist's task of encoding discipline knowledge in the form of workflow-driven ontologies and presents process knowledge in the form of MBWs. MBWs are abstract workflows that capture the steps or processes for generating a result without execution details. Abstract workflows can be used by the scientist to refine the concepts and relationships captured in the WDO, and they can be used ultimately to generate executable workflows.

The WDO-It! tool targets domain experts as the primary users. The overarching goal is to diminish the need for the use of ontologists in the process of initially specifying ontologies and workflows. It assumes that the ontology is dynamic, that there are multiple ways to produce an outcome, and that the user wants to analyze his or her choices to determine the choice that best suits the circumstance. An experimental version of WDO-It! has been developed as a proof of concepts and WDOs have been created in several domains, including gravity data, seismology, and radar signal processing.

Most of the state-of-the-art scientific workflow development and execution environments provide some kind of mechanism with the purpose of enabling scientists to specify and execute workflows. Each environment has its own strengths and ISI Pegasus appears to have the most sophisticated solution to support the scientist in specification of workflows. For example, Pegasus has an abstract way of representing workflows. Moreover, the system allows its users (not necessarily scientists) to use multiple strategies to create abstract workflows including the direct specification of workflows in Pegasus, the use of Chimera to specify logical workflow descriptions, and use of the Composition Analysis Tool (CAT) [6] as an intelligent workflow editor.

Kim et al. describes the users of CAT as "unsophisticated" end-users because it assumes that the workflow that is generated is the desired workflow. Indeed, there could be a number of instances in which this is certainly the case. It is important to note that the WDO-It! process of creating ontologies is integrated with the MBW validation process and that MBW corrections are performed in the ontology. Despite these differences, the systems complement each other. For example, scientists can use WDO-It! to create the ontologies used by CAT to support the creation of abstract workflows at the same time that CAT AI planning can be used to enrich the workflow-driven ontology. Nevertheless, MBWs should be able to be translated into Pegasus abstract workflows enabling the sharing of process knowledge between the systems.

References

[1] G. O. Consortium. Creating the Gene Ontology Resource: Design and Implementation. http://www.genome.org/cgi/content/full/11/8/1425.

[2] E. Deelman and et al. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming Journal*, 13(3):219–237, 2005.

[3] A. Q. Gates, G. R. Keller, F. Salcedo, P. Pinheiro da Silva, and L. Salayandia. The Gravity Data Ontology: Laying the Foundation for Workflow-Driven Ontologies. Technical Report UTEP-CS-07-44, University of Texas at El Paso, 2007.

[4] N. Guarino. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In *Proceedings of SCIE*, pages 139–170, 1997.

[5] 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*, 34(Web Server issue):W729–W732, 2006. doi:10.1093/nar/gkl320.

[6] J. Kim, M. Spraragen, and Y. Gil. An Intelligent Assistant for Interactive Workflow Composition. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI2004)*, Madeira, Portugal, 2004.

[7] B. Ludašcher and et al. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, 2005. Special Issue on Scientific Workflows.

[8] D. Martin and et al. OWL-S: Semantic Markup for Web Services. Technical report, W3C, 2004. Member Submission.

[9] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C), December 9 2003. Proposed Recommendation.

[10] N. F. Noy, M. Sintek, S. Decker, M. Crubézy, R. W. Fergerson, and M. A. Musen. Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[11] N. W. Paton, R. Stevens, P. Baker, C. A. Goble, S. Bechhofer, and A. Brass. Query Processing in the TAMBIS Bioinformatics Source Integration System. In *Proceedings of 11th International Conference on Scientific and Statistical Database Management*, pages 138–147, Cleveland, Ohio, USA, 28-30 July 1999.

[12] L. Salayandia, P. Pinheiro da Silva, A. Q. Gates, and A. Rebellon. A Model-Based Workflow Approach for Scientific Applications. In *Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling*, Portland, OR, October 2006.

[13] 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.

[14] C. Wroe, R. Stevens, C. Goble, A. Roberts, and M. Greenwood. A suite of DAML+OIL ontologies to Describe Bioinformatics Web Services and data. *International Journal of Cooperative Information Systems special issue on Bioinformatics*, March 2003.