A Model-Based Workflow Approach for Scientific Applications

Leonardo Salayandía, Paulo Pinheiro da Silva, Ann Q. Gates, Alvaro Rebellon
The University of Texas at El Paso, Computer Science Department
El Paso, TX, 79902, USA
{leonardo, paulo, agates, arebellon}@utep.edu

Abstract

Productive design of scientific workflows often depends on the effectiveness of the communication between the discipline domain experts and computer scientists, including their ability to share their specific needs in the design of the workflow. Discipline domain experts and computer scientists, however, tend to have distinct needs for designing workflows including terminology, level of abstraction, workflow aspects that should be included in the design. This paper discusses the use of a Model-Based Workflow (MBW) approach as an abstract way to specify workflows that conciliate the needs of domain and computer scientists. Within the context of GEON, an NSF cyberinfrastructure for Earth Sciences, the paper discusses the benefits of using a Gravity Map MBW generated from an ontology about gravity. The Gravity Map MBW is based on terms derived from the gravity ontology that was developed by geophysicists; it does not include some of the workflow properties that tend to make workflow specifications look too complex for discipline domain experts to understand; and it provides a framework for developing strategies to derive executable Gravity Map workflow encodings with only limited interaction from computer scientists.

1 Introduction

Workflows specify the composition of software services, including data and control flow, to achieve a particular result or complete a task. In the case of scientific applications, the design of a workflow typically requires the involvement of at least two domain experts—one from the scientific field of interest (e.g., a geophysicist or biologist) to specify how scientific products (e.g., maps, graphs, data analysis reports) may be derived from datasets and another from a computer scientist, who understands the process of composing a workflow and encoding the derivation in a format that machines can execute.

Productive design of scientific workflows often depends on the effectiveness of the communication between the discipline domain experts and computer scientists—in particular, on their ability to clarify and reconcile their specific needs in the design of the workflow. Because domain experts and computer scientists have distinct terminology to describe workflow elements, including requirements, effective communication is a challenge. For instance, domain experts may base their workflow descriptions on objects of complex types that the computer scientist may not know how to translate to primitive types that are supported by executable workflow languages such as OWL-S [1] and MoML [2].

A domain expert's workflow description is often more abstract than a computer scientist's encodings of a workflow. Additional communication problems arise when the domain expert is expected to understand and further refine workflow specifications prepared by computer scientists. At the same time, computer scientists need to understand the entailments of the domain expert's abstract workflow if computer scientists (with the help of software systems) are supposed to translate the abstract descriptions into executable workflows. For instance, domain experts may be concerned with the specification of partially ordered sequences of services even if such sequences of service do not provide a perfect matching between services' inputs and outputs. In this case, abstract specifications may require further refinement by computer scientists to be executed, e.g., the workflows may require additional steps such as translation services to match input and output information from services.

In this paper we discuss the use of a **M**odel-**B**ased **W**orkflow (MBW) as a means to increase productivity during the design of workflows in support of scientific applications. Following the reasoning from the **D**omain-**S**pecific **M**odeling (DSM) community [3], MBW is also about using a level of abstraction for modeling workflows that is consistent with the target domain, and then using such models (at best) to automatically generate executable workflows, that is, workflow implementations, or (at least) to guide the development of workflow implementations. In this paper we focus on the latter. We present the **W**orkflow-**D**riven **O**ntology (WDO) approach[1] to describe the domain and how WDOs can be used to create MBWs. In a scientific domain with the WDO approach in combination with the service-oriented paradigm, we claim that we diminish the intervention of computer scientists on the software development process by providing tools for domain-experts to produce specifications using the expert's discipline-specific terminology that the computer scientist can employ to create the service-oriented modules necessary to achieve the intended results.

The remainder of this paper is organized as follows. Section 2 describes the technologies involved in representing and executing scientific workflows. Section 3 presents our approach for building model-based workflows and is exemplified through a use case. Section 4 discusses further benefits of model-based workflows when compared to approaches to develop scientific workflows. Section 5 summarizes the paper and identifies future work.

2 Background

Service Oriented Architectures (SOA), in combination with *scientific workflows* and *ontologies* are being used in efforts such as GEON to create cyberinfrastructure [4] that will provide the necessary tools to drive the next generation of scientific research. By developing service-oriented components, the scientific community is developing independent and distributed modules of functionality that are accessible and reusable through the web. Service-orientation enhances the design of low-coupled components by hiding implementation details from users and exposing only an interface specification that serves as a contract between service providers and service users. Ontologies are used first as "an explicit specification of a conceptualization" [6]. Later, they are used to support the composition and matching of services.

Scientific workflows are used to specify the composition of such service modules to achieve some complex scientific endeavor. There are many workflow specification languages and execution engines. Here we mention two: MoML and OWL-S. MoML or the **Mo**deling **M**arkup **L**anguage is the language used by the Kepler Scientific Workflow engine [5] and 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. With the Semantic Web as its basis, OWL-S [1] is a web service ontology that is based on the **O**ntology **W**eb **L**anguage (OWL). OWL-S provides a service provider with constructs to describe properties and the functionality of a service in an unambiguous manner that is interpretable by a machine. 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

[1] http://trust.utep.edu/ciminer/wdo/

how services are composed into a workflow; and the grounding that presents details about how to access the service.

Ontologies are used to describe knowledge about a domain such that its representation can be interpreted and reasoned about by a computer. **W**orkflow-**D**riven **O**ntology (WDO) is an ontology design approach to represent knowledge about scientific domains that thus make them amenable to creating scientific workflows [7]. WDO-specific tools such as the WDO Assistant are used for capturing knowledge. Use cases typically drive the specification of ontologies [8]. In the WDO approach, abstract workflow specifications drive the elicitation and specification of classes and their relationships. For example, domain experts begin the knowledge acquisition process by identifying a *product* and from the *product* identify *methods* that can generate the product. Further, domain experts can identify *data* that are required as input for the identified methods. We claim that abstract WDO-derived workflow specifications are indeed the use cases for WDOs. Such use cases are the basis to create Model-Based Workflows (MBWs) and these are further described in Section 3.2 below. Furthermore, a WDO is an OWL ontology and as such it can be used to represent knowledge that is not workflow-specific, including domain knowledge.

3 Approach

Once a scientist has represented knowledge about a domain of interest by using the WDO approach, the scientist can extract abstract workflow specifications from the WDO that can serve as a guide to implement an application to produce desired information. These abstract workflows are referred to as **M**odel-**B**ased **W**orkflows (MBWs), and are created with the aid of workflow generator assistant software that can interpret the knowledge represented in a WDO. The scientist would identify the information desired from the WDO and the assistant software would then build an MBW to obtain the information based on the concepts and relationships defined in the WDO.

The next section discusses a use case that is used to exemplify the approach, followed by a description of an MBW.

3.1 Use Case

Assume that a geoscientist wants to obtain a *Contour Map* of *Bouguer Anomaly Gravity Data* for a given region of interest. The scientist starts by obtaining a WDO that represents knowledge from the geophysics domain; more specifically about “gravity data.” By using assistant software, the scientist identifies *Contour Map* as the intended information desired, and the assistant software produces as many MBWs as possible from the captured knowledge that identify the abstract steps to produce the map. One of the possible MBWs, referred as the *Gravity Map MBW*, is shown in Figure 1.

To show the relationship to the WDO, the workflow in Figure 1 is divided into two main sections. The left-hand side represents the classes of type *Information* that are associated with the workflow, and the right-hand side represents the classes of type *Method* that are involved in the transformation of the information required to achieve the desired outcome, i.e., a contour map. The left-hand side of the diagram is divided further into three sections: *Product*, *Processed Dataset* and *Data.* The distinction between these classes and their intention is explained elsewhere [7].

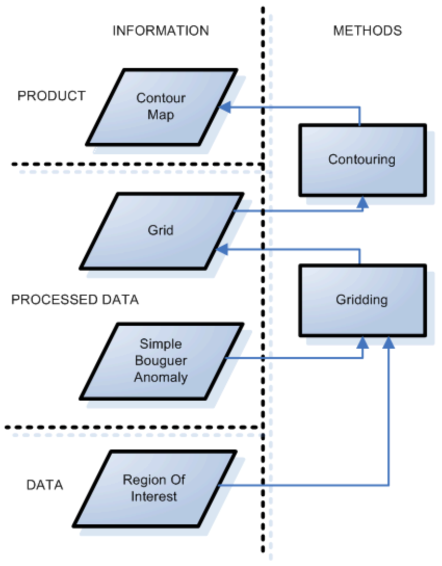

Fig. 1: The Gravity Map MBW generated from the Gravity WDO to produce a *Simple Bouguer Anomaly Contour Map.*

The arrows in Figure 1 shows the data flow of the workflow as the information is transformed starting from information of type *Simple Bouguer Anomaly Contour Map* to *Grid*, and finally to *Contour Map*. The information is transformed through the application of the *Gridding* and *Contouring* Methods, respectively.

3.2 Model-Based Workflows (MBWs)

MBWs are the resulting specifications obtained from a WDO to produce some information desired by the scientist. They are referred to as MBWs because the specifications use the knowledge represented by an ontology, and as a result, the terminology is based on the target domain, not computer science terminology.

Scientific workflows typically involve the sequential transformation of information from a simple information type towards a more complex information type such as an end product. Each step is of the form:

Output Info ← Method (Input Info List)

Output Info defines the type of the information that will result once the *Method* of a step finishes execution. When an *Output Info* type is used as an *Input Info* type in a subsequent statement, it means that the resulting information from this statement is used as input to the subsequent step. Any *Input Info* types that are not bound by previously executing steps require that the user inputs the corresponding type when the execution reaches the given step.

This simple “type-binding” mechanism illustrates the data flow of the workflow specification. The different types of information that will flow through the workflow are: datasets, products, and any other domain-specific concept defined in the WDO to clarify details about the workflow execution.

For example, consider the “Contour Map” use case presented in the previous section. The MBW produced by the assistant software would be as shown in Figure 2. All the concepts in the workflow specification are derived from the Gravity WDO.

Grid ← Gridding (Simple Bouguer Anomaly, Region of Interest);
Contour Map ← Contouring (Grid).

Fig. 2: Model-Based Workflow specification to create a Simple Bouguer Anomaly Contour Map.

Currently we are in the process of formalizing MBW's as an ontology. It is our intension to utilize OWL as a base framework and to have a tight integration between MBW's and our concurrent work on WDO's. The ontology that will be used to represent MBW's will include constructs to specify basic sequential control flow, as well as concurrent control flow to allow workflow specifications with partial order method execution.

4 Discussion

A vision of cyberinfrastructure efforts such as GEON [4] is to provide scientists with tools that would allow them to access and use resources to further their research, education, and professional goals. A short term goal and the focus of this work is to allow domain and computer scientists to communicate better to produce the desired software systems required for scientific endeavors in a more efficient manner. The longer term goal is to provide sophisticated tools that would allow scientists to accomplish their tasks with limited interaction with computer scientists, if any.

Position1: MBWs provide a base for interaction between domain and computer scientists to facilitate communication towards implementing a workflow.

MBWs allow domain scientists to specify their tasks using terminology with which they are familiar, while at the same time assisting computer scientists to understand what needs to be done to implement such specification. After a workflow specification is extracted from the WDO and represented as one or more MBWs, the domain and computer scientists work together to select and refine the MBWs, resulting in an executable specification of a desired system functionality.

The conversion process from an MBW to an executable specification is not straightforward, since the MBW is at a higher level of abstraction and, as a result, will lack details necessary for implementation. For instance, in the *Gravity Map MBW* presented in Section 3.1, the scientist uses the term *Region Of Interest* as input for the *Gridding* method. This requires interaction between the domain and computer scientists to map the abstract data type to one or more primitive data types, e.g., *Double*, *Integer*, and *String*. In one context, a scientist may desire to represent the *Region Of Interest* as two points, i.e., the upper-left and lower-right coordinate values (*Latitude*/*Longitude*) of a rectangular area. The computer scientist may decide to represent the coordinate values with a *Double* primary data type. In a different context, the scientist may decide that the best representation of the *Region Of Interest* may be the name of a county or state. In this case, the computer scientist may choose to map the *Region Of Interest* to a *String* primary data type. In any case, with the help of MBWs, domain experts can specify and refine workflow specifications without specifying a type for the *Region Of Interest* concept or composing a complex type for this concept from the primitive types of a workflow language. Furthermore, existing implementations of the *Gridding* method may only handle *Region Of Interest* represented as a *Latitude*/*Longitude* coordinate value and a *Radial Distance* value. The domain and computer scientists would then have to decide whether to adapt to the existing resource restrictions, or to create additional resources to convert the current needs to match the signature of the existing resources.

Position2: While executable code cannot automatically be generated from MBWs, MBWs guide the code development process.

OWL-S is one executable language that can be used to implement workflows from service-oriented components. Like other executable workflow languages, OWL-S is a sophisticated language that a domain scientist may find discouraging to learn, thus emphasizing the importance of Domain-Specific Modeling approaches. The process of creating an OWL-S workflow or composite service consists of 1) identifying the individual service components to be used in the workflow, and 2) creating the composition process for the workflow.

OWL-S supports a mechanism to create semantic descriptions for service components through "profiles". Following the SOA approach, it is the job of the service provider, who has knowledge of the implementation details of the service component, to provide the description "profile" to the service user, who remains unaware of the implementation details. Once the domain and computer scientists have refined the requirements of the service components to be involved in the implementation of the MBW, the identification of service components is done by matching the requirements to profile descriptions of service components.

The composition process creation follows directly from the composition of methods involved in the MBW, in addition to any intermediary service components that the domain and computer scientists might have identified through the MBW refinement phase. For example, in the contour map use case, the workflow components are the *Gridding* and *Contouring* services, executed sequentially in that order, as described in the MBW.

While tools exist that automatically generate executable scientific workflows from models, e.g., Kepler [5] generates MoML code from a graphical model, such tools do not support Domain-Specific Models, and as a result, lack the consequent benefits of DSM.

Position3: MBWs open doors to additional work that will eventually result in scientists being able to produce workflows with only limited interaction from computer scientists.

Additional complementing work can facilitate the workflow generation process for the scientist. One area that shows promise is *preferences* [9]. Preferences are useful whenever a user has to make a decision, and is an approach that can be used to filter through potentially many options. Preferences may apply both at the model level, as well as at the implementation level. For example, in the contour map use case, the scientist has to decide what is the best representation of the *Region Of Interest* for the context at hand. Once this decision is made, it can be documented as a preference to automate a similar decision for future development in the same context. Similarly, preferences can be captured for the decisions made by the computer scientist that map abstract information types to primary data types. The combination of preferences at all levels of abstraction brings the MBW approach closer to the ideal situation of automating code generation from domain-modeling.

5 Summary

This paper introduced the use of Model-Based Workflow (MBW) approach to facilitate the design of scientific applications. Derived from Workflow-Driven Ontologies built by domain experts, MBWs are described in terms that the experts can understand. Thus, domain experts can be more active in the process of improving workflow specifications and less dependent on their ability to communicate to computer scientists. Although MBWs are very abstract with respect to their implementations, they can still be used as a framework for computer scientists to build executable workflows.

Previous work on expert systems has dealt with the problem of communicating domain knowledge to computer systems. Liou [10] breaks the expert system development effort into four primary tasks: acquiring knowledge from experts; representing that knowledge in a computer-readable form; implementing a prototype of the system; and verifying and validating the system. Even though our goal is not to develop expert systems but to develop scientific workflows, the tasks involved in expert system development contain some parallelism to our work. Our concurrent work on WDO's [7] addresses the issues of knowledge acquisition and knowledge representation through the use of OWL ontologies. WDO's also contemplate validation by allowing domain experts to review and provide feedback about the workflow generation process. The work discussed in this paper deals with prototype building based on the captured domain knowledge. Finally, other work on property specification and runtime monitoring of properties [11] complements the task of verification.

6 Acknowledgements

This work is funded in part by the National Science Foundation (NSF) through grant NSF EAR-0225670. The authors would like to thank Dr. Randy Keller from the Geological Sciences Department at the University of Texas at El Paso for his valuable input in developing the gravity data ontology. Finally, the authors would like to acknowledge the recommendations of two anonymous referees.

7 References

[1] "OWL-S: Semantic Markup for Web Services", *The OWL Services Coalition*, December, 2003.

[2] E.A. Lee, and S. Neuendorffer, "A Modeling Markup Language in XML – Version 0.4", *University of California at Berkley, Technical Report ERL-UCB-M-00-12*, March 2000.

[3] DSM Forum, http://www.dsmforum.org/, August 2006.

[4] The Geosciences Network: Building Cyberinfrastructure for the Geosciences, http://www.geongrid.org/, July 2006.

[5] B. Ludäscher, I. Altintas, C. Berkley et al., "Scientific workflow management and the Kepler system", *Concurrency and Computation: Practice and Experience, Special Issue on Workflow in Grid Systems*, 18(10):1039-1065, 2005.

[6] T.R. Gruber, "A Translation Approach to Portable Ontology Specification", *Knowledge Acquisition* 5(2):199-220, 1993.

[7] L. Salayandia, P. Pinheiro da Silva, A.Q. Gates, and F. Salcedo, "Workflow-Driven Ontologies: An Earth Sciences Case Study", *University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-06-38*, August 2006.

[8] N.F. Noy, and D.L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology", *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05*, March 2001.

[9] M. Bienvenu, and S. McIlraith, "Specifying and Generating Preferred Plans", *Seventh International Symposium on Logical Formalizations of Commonsense Reasoning*, May 2005.

[10] Y.I. Liou, "Knowledge Acquisition: Issues, Techniques, and Methodology", *Proc. 1990 ACM SIGDBP conference on trends and directions in expert systems*, pp. 212-236, 1990.

[11] A.Q. Gates, S. Roach, I. Gallegos, O. Ochoa, and O. Sokolsky, "JavaMac and Runtime Monitoring for Geoinformatics Grid Services", *Proc. 10th IEEE International Workshop on Object-oriented Real-time Dependable Systems*, February, pp.125–136, 2005.