Semantically Resolving Type Mismatches in Scientific Workflows

Kheiredine Derouiche and Denis A. Nicole

School of Electronics and Computer Science,
University of Southampton,
Highfield, Southampton, SO17 1BJ, UK
{kd05r,dan}@ecs.soton.ac.uk

Abstract. Scientists are increasingly utilizing Grids to manage large data sets and execute scientific experiments on distributed resources. Scientific workflows are used as means for modeling and enacting scientific experiments. Windows Workflow Foundation (WF) is a major component of Microsoft's .NET technology which offers lightweight support for long-running workflows. It provides a comfortable graphical and programmatic environment for the development of extended BPEL-style workflows. WF's visual features ease the syntactic composition of Web services into scientific workflows but do nothing to assure that information passed between services has consistent semantic types or representations or that deviant flows, errors and compensations are handled meaningfully. In this paper we introduce SAWSDL-compliant annotations for WF and use them with a semantic reasoner to guarantee semantic type correctness in scientific workflows. Examples from bioinformatics are presented.

1 Introduction

Scientists often utilize computational tools and information repositories to conduct their experiments. Such resources are being made available with programmatic access in the form of Web services. This e-Science approach enables scientists and researchers to work in collaboration. Grid computing builds infrastructures for e-Science to support global distributed collaborative efforts [1]. Research and development efforts within the Grid community have produced protocols, services, and tools that address the challenges of the field. The Globus Toolkit [2] is an open source set of services and software libraries that supports Grids and Grid applications. UNICORE [3] is a system that offers a Uniform Interface to Computing Resources; it defines a layered Grid architecture consisting of user, server and target system tier. gLite [4] is a lightweight Grid middleware developed as part of the EGEE that provides a full range of basic Grid services available for different scientific areas. Scientists are ultimately interested in tools that allow them to bring together the power of various computational and data resources, by developing and executing their own *scientific workflows.* Resources are supplied by third parties and as such the operations provided are often incompatible with each other. Resolving resource mismatches requires the designer's intervention, which can be a difficult and a

R. Meersman, Z. Tari, P. Herrero et al. (Eds.): OTM 2007 Workshops, Part I, LNCS 4805, pp. 125–135, 2007.

time-consuming task for scientists. Another major problem is the inefficient handling of failed workflows. Such complexities should be hidden by the scientific workflow system from the user.

Web services provide the basis for distributed, service-oriented systems. Web service standards such as WSDL provide syntactic descriptions of Web services functionalities using XML Schemas to describe component types. They fail to capture the semantics of complex scientific data. In this paper we propose an approach that integrates semantics into a standard industrial workflow management system, thus allowing the automatic detection and resolution of service mismatches in workflows at design time.

The paper is organized as follows: In Section 2 we provide a general overview of different workflow management systems. In Section 3 we briefly survey the enabling technologies for Semantic Web services. In Section 4 we describe the semantic annotations used to verify the compatibility of services during workflow composition. In Section 5 we present our prototype tool and how it is used to detect and resolve mismatches. In Section 6, we compare our work with existing approaches. Finally, in Section 7 we close the paper by discussing our ongoing work and future directions.

2 Scientific Workflows

Scientific workflows are becoming an important mechanism for scientists to combine scientific data management, analysis, simulation, and visualization tasks. Scientific workflow's characteristics and requirements partially overlap those of *business workflows.* A detailed comparison, however, reveals some significant differences. Business workflows operate on data that is usually stored and managed in databases, e.g. as SQL tables. On the other hand, scientific workflows operate on large, complex, and heterogeneous data. Scientific data is typically stored as large data files encoded in different formats specific to a particular scientific field, e.g. the FASTA format [5] used in bioinformatics to represent protein sequences. These data files maybe indexed in SQL databases for management purposes. Scientific workflows can be computationally intensive, and can produce complex data that is reused in other workflows. Furthermore, business workflow modeling and execution approaches often focus on control-flow and events, whereas scientific workflow systems tend to have execution models that are much more dataflow-driven.

Several business environment workflow technologies have been developed to support effective management of organizational processes. Efforts involved process modeling, and workflow implementation and automation. Business Process Execution Language for Web Services (BPEL4WS) 1.1 [6] is emerging to be an important standard for workflow definition. It forms the basis of the forthcoming WS-BPEL 2.0 OASIS standard [7]. BPEL can be adapted for scientific and Grid services orchestration; its limitations can be overcome by supporting standard technologies such as WS-* specifications [8] [9].

There are several projects that aim to address different aspects of scientific workflows. Taverna [10] provides a graphical interface for biologists and bioinformaticians to build and execute scientific workflows in the Grid. It also supports concurrency, making it suitable for tasks handling concurrent processing.

Windows Workflow Foundation (WF) [11] is a Microsoft technology, part of the .NET Framework 3. The technology allows developers to define, execute, and manage workflows. WF supports two types of workflows: sequential and state machine. Workflows in WF comprise *activities,* typically implemented in a common language runtime (CLR)-based programming language such a C# or Visual Basic®. WF includes a set of general-purpose activities that cover most control flow constructs. WF provides the developers with the ability to develop custom activities to solve their domain-specific problems. Workflows can be designed using a visual designer hosted in Visual Studio through a set of extensions. The workflow structure can be alternatively declared in XAML, a new XML-based language. Although WF is marketed as a tool for designing solutions to business problems, it can be easily leveraged to develop workflows in scientific environment [12].

3 Semantic Web Services

Web services technologies aim to provide reliable, ubiquitous software interoperability across platforms, across networks, and across organizations. Current standard technologies for Web services such as the Web Services Description Language (WSDL) [13] provide only a syntactic-level description of their functionalities. Web services can be published and discovered through UDDI descriptions, offering human oriented metadata that describes what the Web services does, and which organization developed it. Early in 2006 IBM, SAP, and Microsoft discontinued the UDDI Business Registry (UBR) project. The vendors are continuing the support of UDDI standards in their products and services, e.g. Microsoft includes UDDI services in Windows Server 2003. Web services can be invoked using common communication protocols such as SOAP. However, the lack of machine readable semantics necessitates human intervention for automated service discovery and composition, thus restricting their usage in complex business domains.

Semantic Web services technology aims to enable the automation of service discovery, composition, and invocation by augmenting Web services with rich formal descriptions of their capabilities. The concept was proposed around 2001 [14], and the field includes substantial efforts, such as the Web Ontology Language for Services (OWL-S) [15], the Web Services Modeling Ontology (WSMO) [16], and Semantic Annotations for Web Service Description Language (SAWSDL) [17].

4 Semantic Web Service Annotations

4.1 Semantic Annotations for Web Service Description Language

SAWSDL is a set of standards produced by the World Wide Web Consortium (W3C). It is primarily based on the earlier work on WSDL-S [18]. It defines extension attributes that can be applied to elements in both WSDL and XML Schema in order to annotate WSDL interfaces, operations and their input and out messages. SAWSDL semantic annotations are agnostic to the ontology or mapping language used, as long as all the concepts can identified with URIs. SAWSDL provides two basic semantic annotation constructs, Model References, and Schema Mappings.

There exist several tools and APIs that support SAWSDL specifications. SAWSDL4J [19] is one such API implemented in Java allowing the development of SAWSDL based applications. It extends the WSDL4J API for WSDL1.1. Woden4SAWSDL is a WSDL 2.0 parser, based on Apache Woden. Semantic Tools for Web Services by IBM alphaWorks are semantics-based eclipse plug-ins for Web service discovery, and composition. The Web services are annotated using semantic annotations from ontologies in WSDL-S format. The tool infers the ontological similarities of the semantic annotations associated with Web service descriptions. SAWSDL efforts are based on the WSDL-S approach. Radiant [20] is an eclipse plug-in that supports the creation and publication of SAWSDL service interfaces. It also allows adding annotations to existing service descriptions in WSDL through a graphical interface. WSMO Studio is an open source environment for WSMO; it features an SAWSDL editor for adding semantic annotations to WSDL documents.

4.2 Model References

SAWSDL introduces the attribute *modelReference*, a semantic model reference from elements in WSDL or XML Schema to concepts in a semantic model (usually an ontology or taxonomy) via URIs. Model references can be used on WSDL interfaces, operations, message parts, and on XML Schema elements or types. Model references can have many uses, they can provide a classification of a WSDL interface, what a WSDL operation does, and define the semantics of the inputs and outputs of WSDL operations.

XML Schema describe the content of a WSDL message. They define elements associated with a message of a WSDL operation. Operations with parameters of primitive data types such as double or string can have different meanings, since such types tell very little about a the functionality of usage associated with an operation using that type. Model reference annotation associates a semantically defined concept in, for example an OWL ontology, with the corresponding unit of structure in XML Schema. Allowing such annotations can provide value by helping verify type compatibility between operations of connected services. Section 5.1 provides a more detailed description on how the annotations are used to achieve type verification semantically.

4.3 Schema Mappings

The extension attributes *liftingSchemaMapping* and *loweringSchemaMapping* are used to address post-discovery issues in using a Web service. These annotations define a mechanism for specifying the structural mapping of XML Schema types to and from an ontology; such mappings can be used during invocation, particularly if mediation is required.

Lifting schema mappings specify how XML Schema types for WSDL type definitions are transformed to a semantic model, whereas lowering schema mappings define how data expressed in a semantic model are translated to data expressed in an XML document. Both mapping mechanism are agnostic to ontology languages and mapping languages; no restriction exists over the languages that can be used. Section 5.2 describes how the schema mapping annotations are used at runtime to resolve structural mismatches between semantically matched types.

5 Semantic Annotations in Windows Workflow Foundation

5.1 Semantic Parameter Binding in Scientific Workflows

Scientific workflows can be regarded as data-driven workflows, where structured activities whose parameters are compatible are connected using data links. However, this compatibility is realized on a syntactic level only, if the service descriptions are augmented with semantic annotations the compatibility will be ensured at the semantic level as well.

In our approach we consider SAWSDL annotations to ensure parameter compatibility in scientific workflows. The specifications build on existing Web services standards using only extensibility elements. The annotation mechanisms are independent of the semantic representation language. Model references are used to annotate WSDL components and type definitions. Annotations for interfaces and operations provide a high level description of the service capabilities. These annotations are mainly intended to be used in service discovery, matching, and composition. In our approach, we focus on message and type annotations, which provide semantic descriptions on the types of operation parameters in scientific workflows.

In WF, the concept of data links between activities is implemented in the *ActivityBind* class. This class allows the flow of data from one activity to another within a workflow, and it is achieved through binding activity members, such as fields or properties. The mechanism used to validate the data binding of activity properties relies on the assignability of their runtime types. One of the activities that WF supports is an activity for invoking Web services. Web service parameters are exposed as properties that need to be bound to properties within the workflow or properties of other activities. In order to connect parameters in a semantic way, we need to overcome the limitation of the WF approach of data binding validation. To realise the semantic binding of service parameters, we introduce a Semantic Web service to the WF activity library. Using the semantic annotations of parameter types in the Web services, we can automate the binding process, and ensure that connected parameters are semantically compatible at design time.

Binding parameters semantically is based on reasoning over the ontological concepts associated with parameter types. The reasoning process can perform inferences leading to the recognition of semantic compatibility despite syntactic differences. By exploiting the hierarchical structure of ontologies, the reasoning mechanism can differentiate between two types of relations, equivalence and subsumption. If no compatible possible binding is found for a particular parameter, then it has to be manually bound.

Binding connects the parameters of composed Web services, e.g. serviceA and serviceB. If the input parameters of serviceB require their values from the output parameters of serviceA, the user has to manually bind the appropriate parameters between the two services to enable the flow of data in this part of the workflow. If some parameters are syntactically different, the user is not allowed to connect them. Our mechanism will automatically bind an input parameter of serviceB to the semantically compatible output parameter of serviceA. Semantic compatibility is defined by the inferred relations resulting from the reasoning over associated semantic

concepts. If serviceB's input parameter is semantically equivalent to, or a subconcept of serviceA's input parameter, then a binding is established between the two.

5.2 Parameter Mapping at Execution Time

Model references operate at the semantic level to ensure compatible parameters are correctly connected. In WF workflows, the task was accomplished using semantic reasoning in order to automate the bindings of parameters. Compatible semantic concepts can have syntactically different serializations. In order to resolve structural mismatches between compatible parameters, the corresponding ontological concepts need to be grounded to concrete data types. SAWSDL enables the annotation of the type definitions of a Web service with schema mapping extension attributes. The *liftingSchemaMapping* attribute defines how an XML data is transformed to semantic data. On the other hand, *loweringSchemaMapping* attribute defines how data in a semantic model is transformed to XML instance data.

When the WF workflow is executed, the appropriate schema mappings of semantically connected parameters are used to resolve the type mismatch. Between composed services, serviceA and serviceB, a set of parameters are semantically bound. At execution time the evaluated value of serviceA's output parameter is serialized to the defined XML data and translated to the corresponding semantic data. After a successful execution the semantic data is mapped to XML data that conforms to the XML Schema definition of serviceB's input parameter.

5.3 Integration and Implementation

WF's extensibility features allows the development of custom activities to solve domain-specific problems. We developed a new WF activity, called the Semantic Web service (SWS) activity, which allows the semantic description of Web services using SAWSDL documents. By integrating the SWS activity into the WF library, we allow the composition of Semantic Web services and the semantic binding of connected parameters. The implemented SWS activity extends the out-of-the-box Web service activity. It can be generated using SAWSDL documents, as well as WSDL documents.

The .NET framework provides standard .NET libraries for representing, manipulating, reading and writing WSDL 1.1 documents. There is no current support for WSDL 2.0 specifications, so we had to implement the SAWSDL specification for WSDL 1.1 instead of WSDL 2.0. The SAWSDL specification was supported by an implementation of an API that extends the provided WSDL 1.1 library. It currently has full support for all SAWSDL specifications for WSDL 1.1, including model reference annotations for WSDL components, such as operations and messages, as well as XML Schema type definitions, such as XML elements and complex types.

SAWSDL does not restrict the annotation mechanism to a specific ontology representation language. OWL or RDF are two W3C recommended standards, widely used as representation languages for ontologies, and by adopting these standards we gained access to a wide range of existing domain models e.g. life sciences and healthcare. Most importantly, annotation with semantic concepts allows performing semantic reasoning on associated types to infer the compatibility of connected

parameters. Our choice also gave us access to Jena, an open source Semantic Web framework for Java, providing a well supported API that fully supports OWL and RDF. The framework has various internal reasoners, but also provides support for external reasoners such as the Pellet reasoner. A few .NET libraries do exist for the Semantic Web such as SemWeb and Redland libraries, they provide, however, only a partial support. In order to integrate Jena's and Pellet's API into our C# implementation of the SWS activity, we used IKVM. IKVM is an implementation of Java for the .NET framework; it includes a Java Virtual Machine implemented in .NET, a .NET implementation of the Java class libraries, and tools that enable Java and .NET interoperability. IKVM provides a static compiler that converts Java API to .NET Common Intermediate Language (CIL), producing .NET Dynamic-Link Libraries (DLL), thus giving access to the needed Jena features. The semantic binding mechanism connects Web service parameters at design time. Semantic annotations associated with the parameter types are used in Jena to load the appropriate semantic models; enabling inferencing using the Pellet reasoner.

SAWSDL defines schema mappings that overcome the structural mismatch problem between related semantic models. No restriction exists on the choice of the mapping language. However, to comply with our choice of OWL and RDF, we decided to choose an XSLT and SPARQL combination to support the bidirectional mapping between WSDL XSD elements and OWL concepts. The .NET framework does provide libraries to support XML translations technologies such as the XSLT and XQuery specifications. Semantic data, such as RDF graphs, is queried using SPARQL, which is supported in Jena through a query engine. Using a compiled .NET Jena library, the structural mismatches between semantically connected parameters are resolved by executing the associated mappings at runtime.

5.4 Applying the Semantics to Scientific Workflows

In order to assess the value of the semantic annotations for Web services, and the binding mechanism described here, we apply our proposed approach to Web services from the domain of bioinformatics. A large number of public Web services are available in bioinformatics. For example, the European Bioinformatics Institute (EBI) provides several Web services that provide access to services, such as database retrieval and similarity searches. Using the SAWSDL framework, we will be annotating Web services with semantic concepts from ontologies in the bioinformatics domain. ProPreO is a Proteomics data and process provenance ontology, and it is listed at Open Biomedical Ontologies (OBO).

Most of the key data types in bioinformatics have multiple data representation. The inputs and outputs of most bioinformatics operations are weakly typed. In most cases, parameters are either defined as strings or arrays of strings. Annotating parameter types with semantic concepts from bioinformatics ontologies will provide a strong type system where operations from different Web services can be safely composed.

The workflow in Figure 1 is intended to perform a similarity searches over biological sequences. It finds similar sequences to a given DNA sequence. It first retrieves the DNA sequence from the DDBJ database[1], and then it searches for similar

[1] http://www.ddbj.nig.ac.jp

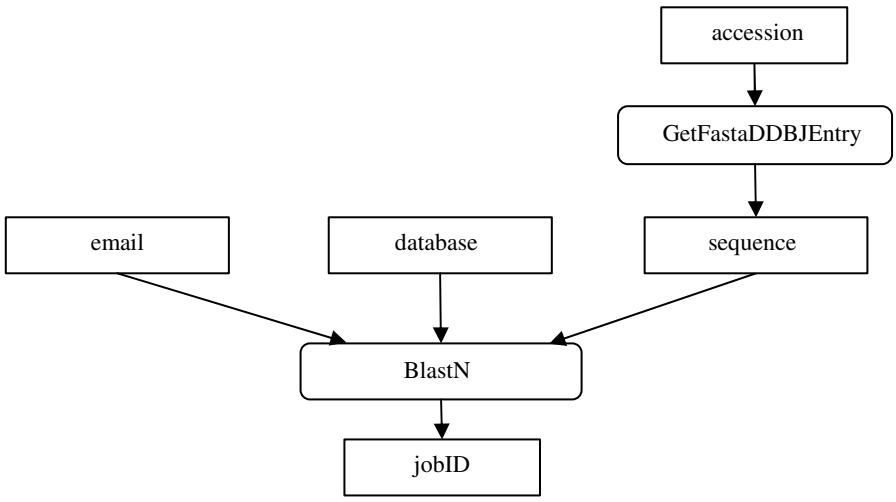

Fig. 1. Automatic binding in a bioinformatics workflow

sequences using *Blast*. The *GetFastaDDBJEntry* is an operation from the *GetEntry* Web service. It takes an *accession* number as input and returns the retrieved DNA *sequence* of FASTA format from the database. This *sequence*, *email* and *database* are inputs to invoke the *BlastN* operation from the *WSWUBlast* Web service, which returns a *jobID* that can be used to retrieve the aligned DNA sequences. All the parameters of the used operations are of type string, which is a primitive type that tells little about the nature of the parameter. We therefore annotate the DNA *sequence* with the *DNASequence* concept. Using our implemented Semantic Web service activity we build the workflow in Figure 1. Our semantic mechanism attempts to automatically bind the inputs of *BlastN* to compatible outputs of *GetFastaDDBJEntry*. Since the parameters of the two operations are both annotated with *DNASequence*, a data binding is automatically established between them. The bound parameters are both of type string, so no translation is needed since no structural mismatch exists.

In a different scenario, the workflow can be modified to use to *BlastP* operation instead, which finds similar sequences to a given protein sequence. Its input parameter is of type string, and is annotated with the concept *ProteinSequence*. Attempting the binding this time will fail since *ProteinSequence* is not a subclass or an equivalent class of *DNASequence*.

6 Related Work

Several Grid workflow systems have been proposed and developed for defining, managing, and executing scientific workflows [21]. Taverna is the workflow management system for the myGrid project, which target bioinformatics workflows. It uses a modeling language called Simple Conceptual Unified Flow Language (SCUFL). Workflows in Taverna possess input and output data entries that can be

annotated with three types of metadata: a MIME type, a semantic type based on the myGrid bioinformatics ontology, and a free textual description. A prototype extension to the Taverna workbench has been developed in an attempt to detect different kinds of mismatches between connected parameters in workflows. The prototype implements a framework that defines layered ontologies to characterize parameter mismatches and accordingly classifies them into several categories. An abstract mapping approach is also proposed in order to resolve detected mismatches. The approach does not define a practical mechanism that supports the grounding of the semantic annotation to concrete data types to support workflow enactment. The parameter mappings are defined as transformation functions between the connected parameters, instead of annotations associated with the structural type of the parameter and the corresponding semantic type. In Triana [22], data links are checked at design time and connected parameters with incompatible data types are flagged with warning messages. In the Kepler [23] system workflows are viewed as a composition of components called actors. Communication between actors happens through interfaces called ports. An object called a director defines how actors are executed and how they communicate with each other. The system handles Web services and Grid services incorporation into workflows, and eventually their invocation and execution. Kepler supports the mapping of parameters that have a type mismatch, but the handled mismatches are a subset of Taverna's proposed extension. However, these mappings do not make use of SPARQL to query semantic models, and instead rely solely on XSLT and XQuery transformations.

Several efforts studied the applicability of BPEL to semantic workflows and Grid environments. Emmerich et al., [24] present a case study where BPEL is used to define scientific workflows, and ActiveBPEL is used as enactment engine for the BPEL definitions. Dörnemann et al., [25] proposed an approach that extends BPEL specification by introducing a new activity to handle the invocation of stateful services. Custom activities, defined within a BPEL composition, cannot be reused later in other workflow definitions. This makes workflow design a complicated task, and with code repetition it makes the workflow unnecessarily large. On the other hand, WF extensibility allows the definition of custom activities that can be reused across different workflows. It also provides a visual designer that facilitates workflow authoring and manipulation for the user. It allows the user to embed C# or Visual Basic code in the workflow to implement simple actions. Compared to BPEL, WF is a lightweight environment for defining, executing, and monitoring workflows. However, neither BPEL nor WF supports checking the semantic compatibility of data types between composed services within a workflow.

7 Future Work and Conclusions

In this paper we have showed how, using business target workflow solution, we can develop fully qualified scientific workflows. Furthermore, we extended the framework to support Semantic Web services and provided a mechanism that lets users develop scientific workflows with no type mismatches by automating the data binding between composed Web services. We have developed a prototype implementation for the approach, and it is executable through Microsoft's Visual

Studio environment. Further optimizations are possible for the approach, which are subject to further research.

Further development of this technology will allow us to ensure that workflows are structured with appropriate compensations and exception handling to minimize wasted computation in failed (deviant) workflows.

References

1. Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1998)
2. The Globus Alliance. http://www.globus.org/
3. UNICORE. http://www.unicore.org/
4. EGEE > gLite. http://glite.web.cern.ch/glite/
5. FASTA Format Description. http://www.ncbi.nlm.nih.gov/blast/fasta.shtml
6. Andrews, T., Curbera, F., et al.: Business Process Execution Language for Web Services Version 1.1. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf
7. Barreto, C., Ballard, V., et al.: Web Services Business Execution Language Version 2.0, Primer, OASIS (2007), http://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm
8. Akram, A., Meredith, D., Allan, R.: Evaluation of BPEL to Scientific Workflows. In: CCGRID, pp. 269–274. IEEE Computer Society, Los Alamitos (2006)
9. Tan, K.L.L., Turner, K.J.: Orchestrating Grid Services using BPEL and Globus Toolkit. In: 7th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool (June 2006)
10. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., Li, P.: Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. Bioinformatics 20(17), 3045–3054 (2004)
11. Windows Workflow Foundation (WF). http://wf.netfx3.com/
12. Paventhan, A., Takeda, K., Cox, S.J., Nicole, D.A.: Leveraging Windows Workflow Foundation for Scientific Workflows in Wind Tunnel Applications. In: SciFlow 2006. IEEE Workshop on Workflow and Data Flow for Scientific Applications, Atlanta, GA (2006)
13. Chinnici, R., Moreau, J., Ryman, A., Weerawarana, S. (eds.): Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation (26 June, 2007) www.w3.org/TR/wsdl20
14. McIlraith, S., Son, T.C., Zeng, H.: Semantic Web Services. IEEE Intelligent Systems 16(2), 46–53 (2001)
15. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services, W3C Member Submission (November 2004), http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/
16. Lausen, H., Polleres, A., Roman, D.: Web Service Modelling Ontology (WSMO). W3C Member Submission (2005), http://www.w3.org/Submission/WSMO/
17. Farrell, J., Lausen, H. (eds.): Semantic Annotations for WSDL and XML Schema, W3C Proposed Recommendation (July 05, 2007), http://www.w3.org/TR/sawsdl/
18. Akkiraju, R., Farrell, J., et al.: Web Service Semantics – WSDL-S, W3C Member, Version 1.0. Submission (November 7, 2005), http://www.w3.org/Submission/WSDL-S/

19. SWASDL4J. http://knoesis.wright.edu/opensource/sawsdl4j/
20. Radiant. http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1
21. Yu, J., Buyya, R.: A Taxonomy of Scientific Workflow Systems for Grid Computing. SIGMOD Record 34(3), 44–49 (2005)
22. Taylor, I.J., Shields, M.S., Wang, I., Rana, O.F.: Triana Applications within Grid Computing and Peer to Peer Environments. J. Grid Comput. 1(2), 199–217 (2003)
23. Ludascher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice and Experience 18(10), 1039–1065 (2006)
24. Emmerich, W., Butchart, B., Chen, L., Wassermann, B., Price, S.L.: Grid Service Orchestration using the Business Process Execution Language (BPEL). UCL-CS Research Note RN/05/07 (June 07, 2005)
25. Dörnemann, T., Friese, T., Herdt, S., Juhnke, E., Freisleben, B.: Grid Workflow Modelling Using Grid-Specific BPEL Extensions. In: Proceedings of German e-Science Conference, Baden-Baden (2007)