A Semantic Web Services Based Framework for Enterprise-Wide Collaborative Engineering Design

Xiaorong Xiang Gregory Madey
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
xxiang1@nd.edu, gmadey@nd.edu

Abstract

An engineering design process may involve the efforts of many discipline specific design partners. These partners may be both cooperative and competitive. In this paper, we propose a web services based framework to support this process. This framework enables composition of the distributed work flow for integrating design models from a standard web browser. It also provides a semantic description of web services using semantic web technologies. The goal for this framework is to facilitate integration of models and tools that contain proprietary information in engineering design domain.

1. Introduction

Engineering design processes, such as automobile and aircraft design, involves several partners located at different locations. These partners may be both cooperative and competitive. Successful engineering design requires well-coordinated interactions between individuals or teams in specialized knowledge domain, information exchange, models, and integration to achieve an optimal goal. However, there is a significant part of design models and tools containing proprietary information that cannot be disclosed. Also, these models and tools are normally written in varieties of programming languages and run on different platforms. With web services technologies, these models and tools can be treated as black boxes and run at their original locations.

In the integration process, two techniques are involved: exposing part or all of the functionalities as web services (web service management) and developing an application that uses web services (web services composition). In this paper, we propose a semantic web services based framework that supports the collaborative engineering design process. This framework enables composition of the distributed work flow for integrating design models from a standard web browser. It also allows publishing and management of web services using semantic web technologies. Instead of using the proposed DAML-S and OWL-S standards for the semantic description of web services, we propose an alternative approach that adds one more ontology layer above the WSDL to describe the properties of web services. The goal for this framework is to facilitate integration of models and tools that contain proprietary information in the engineering design domain.

A view of the collaborative engineering design problem domain is presented in section 2. Important existing technologies in semantic web services are described in section 3. The proposed semantic web services based framework and the implementation issues are illustrated in section 4. Conclusions are drawn and future works are described in section 6.

2. Enterprise-Wide Distributed Multilevel Design

In Figure 1, the logical view of the system, subsystem and component optimization modules, represent the distributed multilevel design optimization process. The system consists of two major disciplines: the engineering discipline and the business discipline. The distributed enterprise, business analysis models, and information retrieval services are available online for design, planning, and decision making by end users in the organization. Some of the engineering analysis models are enterprise resources and accessible by all authorized users, while other models may be proprietary to a supplier. Some of the analysis models may be applied at the system level, while others may be applicable at the subsystem or component level.

The design process is often a collaborative process that involves multiple specific design teams in the same orga-

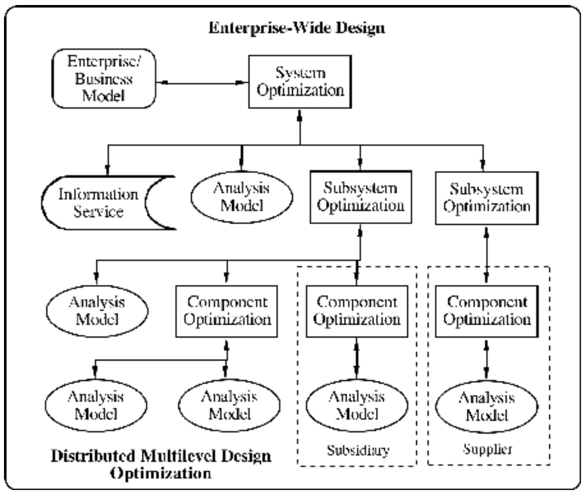

Figure 1. Logic View of Distributed Multilevel Design Optimization

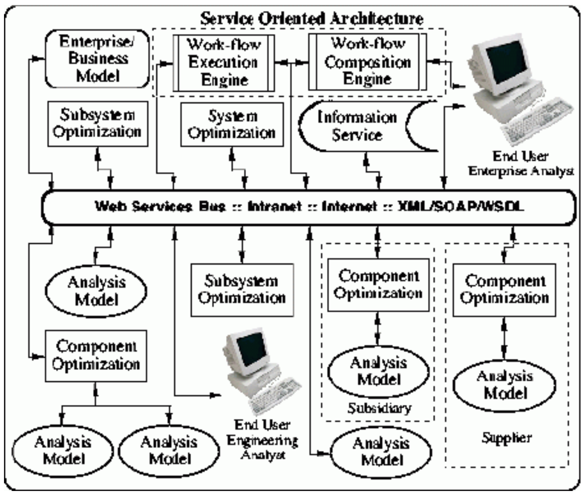

Figure 2. Web Services Deployment for Distributed Multilevel Design Optimization

nization or across the organization's boundary. Integration of these discipline specific models is a daunting challenge. However, the emerging semantic web services technologies can help the integration and decision making process. In Figure 2, each enterprise wide analysis module and information services module are wrapped as web services. They are "plugged" into the Internet and enterprise Intranet. By using open web services standards, existing legacy and new applications, are deployed as online interoperable distributed services. An engineering analyst (designer) and an enterprise analyst are able to access any single web services module. The analysts can compose a collection of individual web services into a composite web service potentially comprised of many optimization, analysis, and information retrieval services. The composite services must be sequenced so that they execute in the proper order, reflecting data dependencies, and solving a coherent enterprise design problem.

One simple example of a collaborative engineering design problem is the Aircraft Concept Size (ACS) problem [4]. This problem involves the preliminary sizing of the general aviation aircraft subject to certain performance contraints. It consists of three disciplines: aerodynamics, weight, and performance. These three disciplines can be represented as three optimization models with clearly defined input and output as shown in Figure 3. They can reside at distributed locations. The enterprise analyst gives the constraints and goals for the aircraft design. Three models need to participate the optimization process with multiple interations with the enterprise analyst.

3. Semantic Web Services Technologies

Fundamental web services technologies, such as WSDL, SOAP, and UDDI, provide e-Business a way to automate and streamline distributed business processes. The combination of web services technologies and the semantic web can facilitate the autonomous and heterogeneous applications, data, and services residing in distributed environments. In order to accomplish this goal, several issues are addressed in semantic web services research.

- **Service description**. In order to enable automatic discovery, invocation, and composition of web services, a computer interpretable description is needed. The proposed standards, such as DAML-S and OWL-S [10], provide an upper ontology for describing not only the input and output but also properties and capabilities of web services.

- **Service publishing and discovering**. In order to let other people use the service, service providers should register their services in a public registry or a private registry inside of an organization. A service consumer needs to discover a service that meets the functional requirements. UDDI [13] provides a standardized method for publishing and discovering information about web services. However, in order to support the automation of the discovery process, the registration information should include both syntactic and semantic description of services.

- **Service composition**. Simple autonomous services are not enough to represent a large application process.

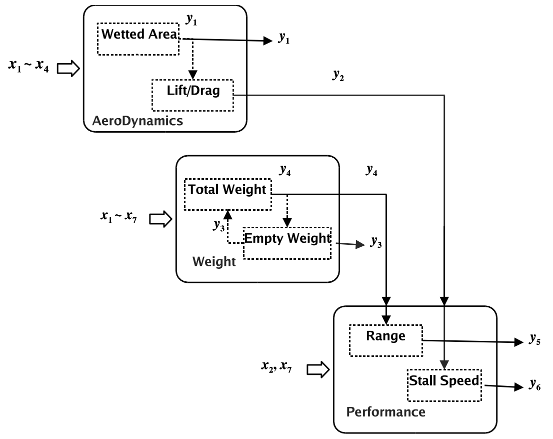

Figure 3. Aircraft Concept Size (ACS) problem (adapted from Gu and Renaud[4])

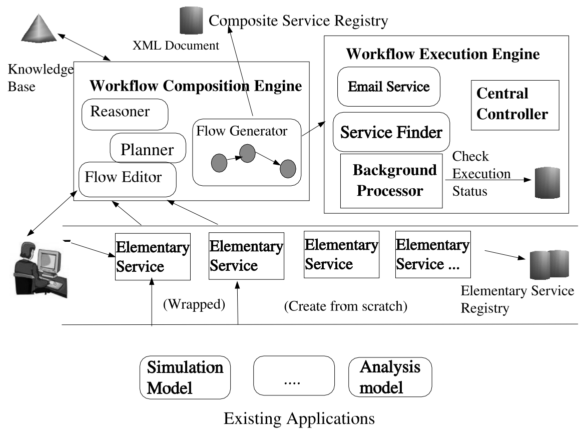

Figure 4. The Semantic Web Service Based Framework Architecture

An approach is needed to generate a composite service based on existing ones. The composition process can be done manually, semi-automatically [12], or automatically [2].

- **Service execution**. Service execution for composition services largely depends on the composition model. Either by transferring the composition model into a particular workflow specification, such as BPEL4WS, and executed on a workflow engine [6], or by representing the composing process using DAML-S specification and executing it by a DAML-S Virtual Machine [12].

Although there are various proposed semantic web services technologies, no widely accepted service representation and deployment standard exists. In the next section, we present our proposed framework that adapts the current existing technologies to facilitate the enterprise wide engineering design process.

4. The Semantic Web Services Based Framework

From the functional perspective, the framework should provide the following components: semantic description of web services, services publishing and discovery, service composition engine, and service execution engine. Figure 4 shows the infrastructure of the framework. The framework participants can be separated into three types: service providers, service consumers (engineering analyst or enterprise analyst), and service composer (enterprise analyst).

Service providers are the entities that offer simple web services. They are responsible for describing their web service by providing values to the properties, and publishing their services. Service consumers may be end users or other web services that invoke a web service: either a simple or composite one. Service composers are responsible for specifying composite services.

In this paper, an *elementary service* is defined as a simple service that does not interact with other services. A *composite service* is defined as a complex process that is composed of multiple *elementary services*.

4.1. Semantic Description of Web Services

DAML-S and OWL-S are two emerging ontologies for describing the capabilities and properties of web services. They can be used to describe not only the elementary services but aslo the composite services. Medjahed et al [7] propose a semantic description method and composablity model for web services. Although the latter approach does not provide the semantic description of composite services, we discover that it is more flexible to add semantic description for elementary services.

Therefore, We adapt the semantic description method and composability model for web services that have been described in [7] and use OWL [9] to specify the web services ontology. We model the ontology using a directed graph. The web service ontology consists of WSDL concepts and extended features to represent the semantic capabilities.

Extending the approach of Medjahed et al [7], we add more syntactic and semantic description for a web service, such as the URI of the WSDL file. Also for the quality definition of web service, we add the availability and the latency of execution the service.

OilEd [8] and Protege [11] are ontology and knowledge-based visual editors. Jena [5] is an open source toolkit used to create semantic web applications. It implements the RDF and OWL language and provides ontology API and inference subsystems. The Jena toolkit can be used to build the knowledge base for web services. Both tools can be used to management the ontology of web services.

4.2. Service Registration and Discovery

Elementary web services can be created with different languages and deployed on various platforms, such as J2EE and .NET. Elementary web services can be separated into several categories, including information input services, information query and data analysis services, computation services, and computing resource management services.

Service providers are responsible for creating and deploying these elementary web services. In the engineering design domain, some elementary web services can be created by wrapping existing proprietary applications.

In addition to elementary web services creation, the service providers also need to register the syntactic and semantic information of their services. The service registration process is shown in Figure 5. The provider needs to create the WSDL, which defines the service's syntax by specifying the structure of the input and output message, along with the service's runtime bindings.

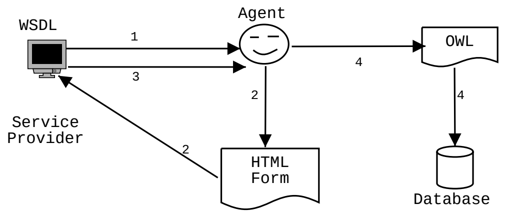

Figure 5. Service Registration Process for Syntactic and Semantic information

1. The service provider starts the registration process by providing the URI of the WSDL file.

2. A software agent on the service registration side extracts the information from the WSDL file. According to the WSDL information, a software agent generates a HTML form that requires the semantic description of service and operations.

3. The services providers can fill out the form to specify the meaning of the WSDL.

4. Then the software agent generates a OWL description for the service and stores it into the database.

Service consumers can discover a service by first identifing the required functionality. Then they access the service category through taxonomies and find the approximate accurate service.

4.3. Service Composition

For a system with a small number of elementary web services, the domain experts can manually define a set of rules describing the relationship among existing services and the application process logic. However, for a large scale system that consists of a large number of services and new emerging services, manually predefining the composability rules is not practicable. With the semantic description of web services, it is possible to define the composability rules for creating a composite service. Medjahed et al [7] identify two sets of composability rules to compare syntactic and semantic properties of web services. We adopt four of their rules for the service composition.

1. Mode composability compares the operation mode.

2. Binding composability checks the communication protocol of interacting services.

3. Message composability compares the number of message parameters, data types, business roles, and units.

4. Operation semantics composability compares the semantics of service operations.

The workflow composition engine in the framework provides the service composer a way to construct a composite service. It is intended to support three composition modes: ad hoc, semi-automated, and automated compositions.

- **Ad Hoc composition**: Under certain situations, the service composer (enterprise analyst) knows the application logic and exact components that should be included in the application process. The service composer can form the composite service manually. The workflow composition engine needs to validate the composite service.

- **Semi-automated composition**: The workflow composition engine lets the service composer create the composition process by presenting all the optional services at each step. The end user makes the decision.

- **Automated composition:** The workflow composition engine lets the service composer provide the input and output information. These information are fed into an AI planner, such as SAPA [3]. The planner returns one plan, multiple plans, or no results to the end user for further decision.

If the service composer is satisfied with the generated workflow, the workflow composition engine assigns an identifier to the workflow. The workflow specification, either represented by an existing language (BPEL4WS for example) or a self-defined XML document, is generated to present the workflow for this task. The specification is stored in a database corresponding to the task identifier. Next, it is sent to the workflow execution engine.

4.4. Composite Service Execution

In the framework, the workflow execution engine consists of a central controller and a background processor [14].

- **Central controller** is implemented by exploiting Java Servlets technologies. It is responsible for handling the synchronous web service invocation for short running jobs such as data input that requires maintain the status of interactions between end users and services.

- **Background processor** is responsible for handling the asynchronous invocation for long running jobs such as the computation services. It monitors the status of all the submitted tasks by checking the database periodically. When it discovers that a workflow is ready for invoking a computation service, it sends a SOAP message to the server.

  This approach allows end users to submit their workflow specifications, input the parameters interactively with services, and disconnect from the system and retrieve all the data later. SAIWS [15], a simple asynchronous invocation framework for web services, is built upon the Apache Axis that can be used for this purpose.

There are two issues involved in the service execution stage need to be considered.

- **Locating an alternative service**

One of the issues in the service execution is that the selected service in the workflow is unavailable or off line temporarily. Then the execution engine invokes the alternative service if there is one defined in the workflow at the service composition stage. Otherwise, the **Service finder** in the execution engine searches the knowledge base to find a alternative service if there is one.

- **Monitoring service execution**

Web services often need long time to complete the execution. Services consumers need to know what the status of their requesting services. For executing an elementary service, a service consumer may want to know if the service is completed at a time point or if an unexpected exception occured. For executing a composite service, a service consumer needs to know where in the process the request is. Monitoring the service execution status is another important issue.

In this framework, the execution status of the web service is recorded in the database. The status of a web service includes: which requests are finished, which request is invoked but not finished at this point, and if any unexepected exception occured for the current invoked request. Service consumers can retrieve the executing status of their requested service from a web browser and decide if they want to stop the execution. After the workflow execution is finished, the **Email service** sends an email to the service consumer to notify of the completion.

5. Conclusion and Future Work

The presented framework is intended to support the E-engineering application, specifically the enterprise wide collaborative engineering design problem. However, it can be easily adapted to support E-science applications, such as distributed scientific simulations. The framework emphasizes several issues for using semantic web services technologies including semantic description of service, services registration and discovery, services composition, and service execution.

More functionalities of web service execution monitoring need to be address in the framework in the future. Currently, the framework allows end users to access the execution status of their requested services. However, service consumers can not change their plans after a composite service is invoked. Also at the current stage, the execution of a composite service is centralized because it is easy to realize. However, a centralized execution model posses scalability, availability, and security problems [1] as the emerging peer-to-peer execution model becomes more attractive.

Acknowledgement. The authors would like to thank Dr. John E. Renaud, who is a professor at the Department of Aerospace and Mechanical Engineering in University of Notre Dame, for his discussions and input on the collaborative engineering design problem.

References

[1] B. Benatallah, M. Dumas, Q. Z. Sheng, and A. H. Ngu. Declarative composition and peer-to-peer provisioning of dynamic web services. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, 2002.

[2] M. Carman, L. Serafini, and P. Traverso. Web service composition as planning. In *ICAPS 2003 Workshop on Planning for Web Services*, 2003.

[3] M. Do and S. Kamghampati. Sapa: A multiobjective metric temporal planner. *Journal of Artificial Intelligence Research*, pages 155–194, 12 2003.

[4] X. Gu and J. E. Renaud. Implicit uncertainty propagation for robust collaborative optimization. In *Proceedings of DETC'01 ASME 2001 Design Engineering Technical Conference and Computers and Information in Engineering Conference*, 2001.

[5] Jena. http://jena.sourceforge.net/index.html.

[6] D. J. Mandell and S. A. McIlrith. Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation. In *Second International Semantic Web Conference (ISWC2003)*, 2003.

[7] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journal*, 2003.

[8] Oiled. http://oiled.man.ac.uk.

[9] OWL. http://www.w3.org/2001/sw/WebOnt/.

[10] OWL-S. http://www.daml.org/services/owl-s/1.0/.

[11] The protege project. http://protege.stanford.edu/.

[12] E. Sirin, J. A. Hendler, and V. Parsia. Semi-automatic composition of web services using semantic descriptions. In *Web Services: Modeling, Architecture and Infrastructure workshop in conjuction with ICEIS2003*, 2003.

[13] Oasis UDDI. http://www.uddi.org.

[14] X. Xiang and G. Madey. A semantic web services enabled web portal architecture. In *International Conference on Web Services (ICWS2004)*, 2004.

[15] U. Zdun, M. Voelter, and M. Kircher. Design and implementation of an asynchronous invocation framework for web services. In *The International Conference on Web Services - Europe 2003 (ICWS-Europe'03)*, 2003.