

# A Semantic Web Services Enabled Web Portal Architecture

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

Web services technologies are emerging as a new approach for supporting e-Science and e-Engineering by providing access to heterogeneous computation resources and integration of distributed scientific applications. In this paper, we present a web portal architecture that enables end users to access the distributed application through the web interface and to compose new tasks by integrating a set of high level web services provided by portal developers. The portal design demonstrates that flexibility, reusability and interoperability can be provided for both end users and portal developers by using semantic web services technologies.

## 1. Introduction

Web services technologies provide e-Business a way to automate and streamline distributed business processes. They are also emerging as a new approach for supporting e-Science and e-Engineering by providing access to heterogeneous computation resources and integration of distributed scientific and engineering applications. Industries provide a set of standards to describe the web services and work flow. The limitation of this approach is that the discovery and composition processes must be manually processed. A computer interpretable description of web services is needed to enable automatic discovery, invocation and composition. Built on the foundation of RDF, RDF Schema, and DAML+OIL, DAML-S provides an upper ontology for describing properties and capabilities of web services to meet these purposes. These semantic web technologies have been employed for exploring the web service automation [3] [5].

## 2. Web Portal Architecture Overview

Portal participants can be separated into three roles: service providers, service requesters, and end users. Portal developers can be service providers and/or requesters. End users are scientists and engineers who are experts in their research domain but may be novices in web service technologies. The purpose of a web portal is to provide end users an approach to access software and to share data and information that reside on a distributed computational environment via a standard web browser. Figure 1 shows the semantic web services enabled web portal architecture.

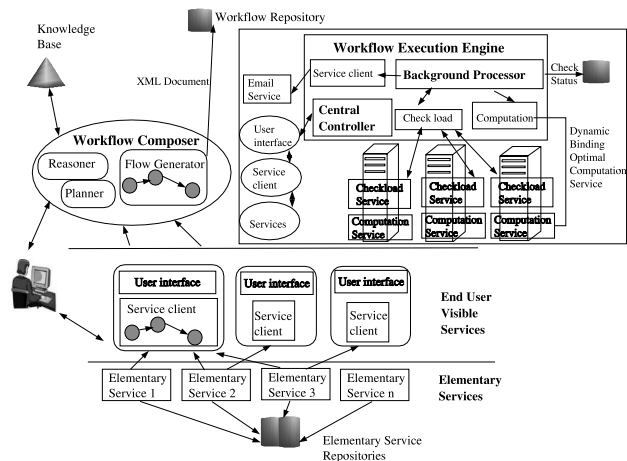


Figure 1. Semantic web services enabled web portal architecture

### 2.1. Elementary Web Services Creation

Elementary web services can be created with different languages and deployed on various platforms, such as J2EE and .NET. Elementary web services provided in the portal

can be separated into several categories, including information input services, information query and data analysis services, computation services, and computing resource management services. Some web services provide end user usable functions, while other web services provide the functions for monitoring and managing the computing resources. These web services can be stored into a central repository or multiple repositories.

## 2.2. End User Visible Services

In addition to the elementary web services, service clients and user interfaces are provided for accessing services that can be used by end users. These are called end user visible services. End user visible services can be complex web services that are composed by multiple elementary services according to a fix application logic. End users can access web services directly or compose new application without knowledge of how to interpret the underlying specification.

## 2.3. Semantic Web Service Description

OWL-S [4], a OWL-based Web service ontology, is proposed as a new version of DAML-S. It is used for description of web services. Jena [2] 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. The portal maintains a knowledge base that contains OWL-S description of existing services.

## 3. Web Services Composition Framework

A task editor, a simple HTML template or sophisticated graphic composition tool, is provided to end users for web services composition. Also the web service composition framework is intended to facilitate the portal developers to integrate new services into the portal for public usage.

### 3.1. Workflow composer

The workflow composer is intended to support two composition modes, semi-automated and automated compositions. **Semi-automated composition:** The composer lets the end user create the composition process by presenting all the optional services at each step. The end user makes the decision. **Automated composition:** The composer lets the end user provide the input and output information. These information are fed into a artificial intelligence planner, SAPA [1]. The planner returns one plan, multiple plans, or no results to the end user for further decision.

If the end user is satisfied with the generated workflow, the composer sends a query to the database and gets a session number as an identifier for the task. The workflow specification, either represented by an existing language (BPEL4WS for example) or a self-defined XML document, is generated to present the workflow of this task. The specification is stored in a database corresponding to the task identifier. Next, it is sent to the workflow execution engine.

## 3.2. Workflow Execution Engine

The workflow execution engine consists of a central controller and a background processor. **Central controller** 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. 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 [6] is a simple asynchronous invocation framework for web services that is built upon the Apache Axis can be used for this purpose.

## 4. Future Work

The presented portal architecture is intended to support two real world applications, an E-Science and an E-engineering application. The portal architecture provides limited service management capabilities to end users. End users can query the execution status of a composed service while it is executing and stop it if necessary. However, end users are unable to modify the plan after it start running. More control capabilities for end users are planning to add.

## References

- [1] M. Do and S. Kamghampati. Sapa: A multiobjective metric temporal planner. *Journal of Artificial Intelligence Research*, pages 155–194, 12 2003.
- [2] Jena. <http://jena.sourceforge.net/index.html>.
- [3] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journal*, 2003.
- [4] Owl-s. <http://www.daml.org/services/owl-s/1.0/>.
- [5] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating daml-s web services composition using shop2. In *In Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, 2003.
- [6] 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.