Wedding the Web: An Example of a Services and Semantics Marriage that Works

ABSTRACT

Web service standards are enjoying widespread adoption. Corporations are using Web standards to describe their application's interfaces and data to make it easier for other systems to consume these products. Using these standards, software clients written in one programming language can access and retrieve information from a server, regardless of the technology (e.g., hardware, operating system, and programming language) used by the server. However, even with the adoption of these Web standards, the programmer writes much of the "glue code" necessary for a particular operation.

In this paper, we illustrate how Semantic Web technologies can be used to discover Web service workflows and automatically generate the "glue code" needed to chain the invocation of these Web services together and execute them. Our simple example shows how a marriage of Web services and Semantic Web technologies can further automate this Web service orchestration process. Our example illustrates a solution that uses domain ontologies to abstract databases and model Web service processes. We use this information to discover Web service workflows and automatically generate code to execute this workflow.

Keywords: Web Services, Semantic Web, workflow, ontologies

INTRODUCTION

Web service standards are enjoying widespread adoption in corporations across many industries. Corporations are recognizing the value of making it easier for other applications to consume their data by using Web standards such as the Hyper Text Transfer Protocol (HTTP), Web addressing, and Extensible Markup Language (XML). Using these standards, software clients written in one programming language can access and retrieve information from a server, irrespective of the technology (e.g., hardware, operating system, and programming language) that the server uses.

However, even with the adoption of these Web standards, problems remain. For example, although XML is mature as a syntax for Web data exchange, current XML technologies do not supply the capabilities provided by more mature technologies like relational database systems. Also, while solutions that aid in Web service discovery (e.g., Universal Description, Discovery and Integration (UDDI)[1]) and invocation (e.g., Web Services Description Language (WSDL)[2]) are emerging, they are far from mature. Similarly, technologies that reason with Web service description files for the purpose of chaining Web services are not available. It is left to the programmer to determine, at design time, which Web services to invoke, the order in which they need to be invoked, and the formatting of information necessary to complete an operation. As a result, the programmer writes much of the "glue code" necessary for a particular computation.

Today, Semantic Web technologies are beginning to emerge on the market with promises of enabling a much faster integration of applications and data. However, for that to happen, Semantic Web technologies must enable solutions to the above-mentioned problems. Web services must be discovered, chained, and invoked automatically, thus relieving the programmer from having to do these steps. Semantic Web standards provide a rich framework for the precise description of data and applications, thereby enabling greater automation in this end-to-end Web service execution process. The World Wide Web Consortium (W3C) proposed recommendation for a standard Web ontology language (OWL [3]) builds on Web technologies including "XML's ability to define customized tagging schemes and RDF's flexible approach to representing data." [4]

Convergence between Web services and Semantic Web technologies is beginning as illustrated by the OWL-Services (OWL-S) [5] effort. OWL-S (formerly DAML-S) is an active research area funded by the DARPA Agent Markup Language Program [6]. OWL-S is an effort to develop a Web service ontology that could be used to describe the properties and capabilities of Web services in unambiguous, computer-interpretable form.

In this paper, we use a simple example to illustrate how Semantic Web technologies can be used to discover Web service workflows, to chain the invocation of Web services, and to automatically generate the "glue code" needed to execute these Web services. Our example shows how a marriage of Web services and Semantic Web technologies can further automate this Web service orchestration process. While our example does not currently use OWL-S, it is an approach that could build upon OWL-S to implement a complete solution. Our example also illustrates that part of the solution can be using ontologies to abstract databases, thus making their data available to Web services.

A SIMPLE EXAMPLE

In this section we describe a simple example that illustrates how Semantic Web technologies can be used to discover workflows of Web services and automatically generate the "glue code" needed to chain the invocation of these Web services together and execute them. We use a Web-based application service like MapQuest© to illustrate our approach and to motivate how one might successfully marry ontologies and Web services together.

The Courtship – Setting the Foundation

In this section we describe the foundational concepts used in our approach. This includes using a domain ontology to capture key domain concepts, a database to provide the instance data to be mapped to the domain ontology, and a Web Service Description Language (WSDL) mapper to capture information about Web services and integrate them into the domain ontology.

Domain Ontology. Legacy data is often contained in a database management system. A domain ontology can be used to represent key concepts from a database. Fields in the database can then be mapped to concepts in the ontology. In this way, the domain ontology becomes an abstraction of portions of the database with the database contents providing instance data for the domain ontology. Figure 1 shows a simple domain ontology. Consider that there is a database table mapped to this domain ontology that provides the name and address of a given business. Figure 2 shows one entry in the database table. In this example, the contents of the "Business Name" column in the database would be mapped to the concept Name in the domain ontology. Likewise, the column "Street Address" and "Zipcode" would be mapped to the domain ontology concepts Street and Zip respectively. Notice that the domain ontology includes concepts that are not included in the database (i.e., Location concept). Our example also includes one data field that is not included in the domain ontology (i.e., "State"). Tools already exist that provide the ability to map relational database tables to ontologies (e.g., [7] [8]). Although our examples use ontologies to abstractly represent the contents of a database management system, this same approach could be used with other data sources such as the output of Web services or structured text files.

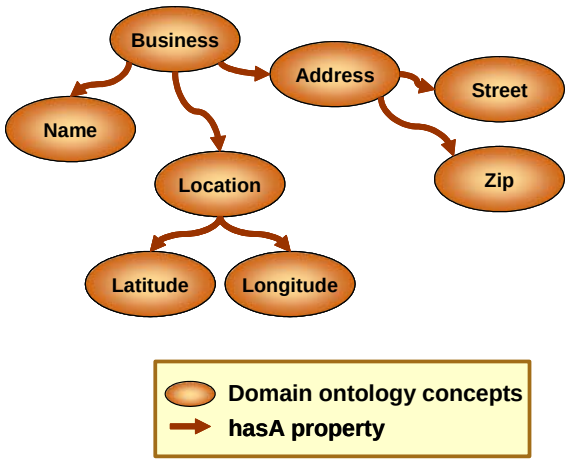

Figure 1. Domain Ontology

Business Name	Street Address	State	Zipcode
…	…	…	…
The MITRE Corporation	202 Burlington Road	MA	01730
…	…	…	…

Figure 2. Sample Database Entry Mapped to the Domain Ontology

WSDL Mapper. Another tool in our approach is a WSDL mapper that performs two key operations. First, the WSDL mapper takes a Web service and augments a given domain ontology with concepts that represent the Web service and with properties that describe the process of using that Web service. The WSDL mapper uses a standard set of properties to define the process for using the Web service. These standard properties include a characterization of the input required for the Web service (isInputTo) and the output generated from the Web service (hasOutput). For example, lets assume that we want to associate with our domain ontology a map Web service that takes a street address and zip code and returns a map of

the locality of that address as an output. Figure 3 shows the augmented domain ontology that results from our WSDL mapper.

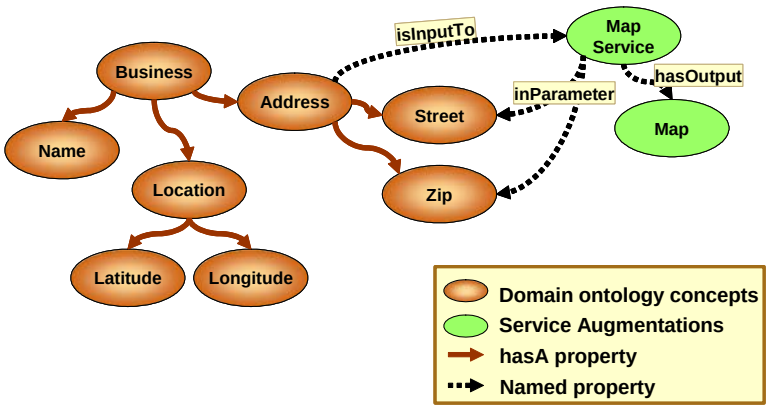

Figure 3. Augmented Domain Ontology

Besides augmenting the domain ontology, the WSDL mapper also uses the Web service to create classes and instances in a Web service ontology. This Web service ontology contains all the information necessary to access and execute that Web service. This includes information on the Web service input parameters, parameter types, access protocols, and Web address (URL). In our example, the Web service ontology would include an instance of a Map Application Service, along with all the information needed to access and execute this Web service captured in associated property values. A simplified version of the portion of the Web service ontology related to the Map Service, and using MapQuest© as the sample Map Service, is shown in Figure 4.

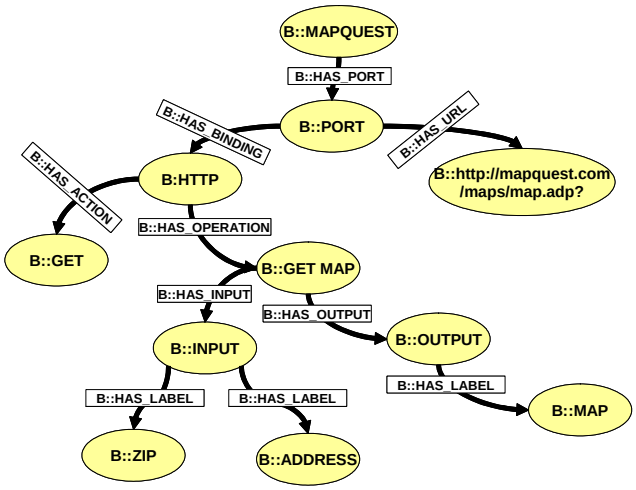

Figure 4. Simplified Web Service Ontology for MapQuest© Map Service

The Marriage – Using ontologies to integrate data and Web services

In this section we describe how we use the foundational concepts described above to discover workflows that satisfy user requests and automatically generate the "glue-code" needed to execute a user-selected workflow. We also show how an ontology can be used as the intermediary to automatically integrate data contained in a data repository with a Web service. We use our simple Web-based map application to illustrate this approach.

Workflow Discovery. We are using an algorithm that uses the augmented domain ontology to discover possible Web service workflows. Our algorithm exploits the standard properties defined by the WSDL mapper to derive Web service workflows. Given a list of input values and a desired output, the algorithm does three things. First, the algorithm creates the set of all the direct paths that lead to the desired output concept where a direct path is the sequence of nodes (i.e., concepts in the ontology graph) that lead to the desired output concept. Second, for each input value, the algorithm creates the set of direct paths that lead to the given input. Finally, the algorithm calculates and returns the intersection of these sets. This algorithm may be defined more formally as follows:

For input list i_n, output v
Find the list of nodes $x_i\{x_1, x_{2,..}\}$ that has direct paths P_k $\{p_1,p_2,..\}$ with v
For each i in i_n, find the list of nodes $y_j\{y_1, y_{2,..}\}$ that has direct paths Q_m $\{q_1,q_2,..\}$ with i
Return $\{x_i, P_k, Q_m\}$ where $x_{i=}y_j$

The workflow discovery algorithm is triggered by a user or application providing an input and a desired output that has been mapped to concepts in the domain ontology. Consider an example that uses the augmented domain ontology shown in Figure 3. Given an input of the business name "The MITRE Corporation" and a desired output of a Map, our algorithm would discover one workflow. The details of how this workflow was derived follows.

Find direct paths to desired output concept:

- x_1\{Map Service\}; p_1 \{Map Service; hasOutput; Map\} //P1 is the path that connects Map Service to Map
- x_2\{Address\}; p_2 \{Address; isInputTo; Map Service; hasOutput; Map\} //P2 if the path that connects Address to Map
- x_3\{Business\}; p_3 \{Business; hasA; Address; isInputTo; Map Service; hasOutput; Map\} //P3 is the path that connects business to map

Find direct paths to given input:

- y_1\{Business\}, Q_l \{Business; hasA; Name; "The MITRE Corporation" isA Name\} //where isA implies is an instance of the concept that was populated by the database (more on this below)

Return discovered workflows

- Return \{Business, p_3, Q_l\} //since Business is in x and y

The algorithm assumes that input values and desired results are mapped to concepts in the domain ontology. In our example, the input value "The MITRE Corporation" and the desired result Map were selected from the contents of the domain ontology. That is, the domain ontology includes an instance of the concept Business that has an associated instance of the concept Name with the value "The MITRE Corporation" that was selected as an input value. The concept Map was selected as a desired output. This example shows that the input provided need not be the specific input parameter required by any given Web service. Rather, our algorithm can discover workflows using concepts and data available in the domain ontology. In our example, the domain ontology includes an instance of the concept Business that has mapped to it an instance of the concept Name with the value "The MITRE Corporation", an instance of the concept Street with the value "202 Burlington Road", and an instance of the concept Zip with the value "01730". Including this instance data in the ontology allows the algorithm to discover the connection between the selected instance of Name and the desired output of Map. This very simple example shows that ontologies may be used to abstractly represent concepts in databases and use the databases to populate instances of these concepts in the ontology. In this way, database contents may be made available to Web services.

The mapping of user inputs to concepts in the domain ontology could be accomplished by using a controlled vocabulary of concepts in the available ontologies. Tools could be developed to assist the user or application in associating the input values and desired outputs to concepts in the available ontologies. Specific approaches to accomplish these mappings are not a focus of this paper.

Now let's consider the impact of using the WSDL mapper to add a Map Point Service and an address Translation Service to the domain ontology. This MapPoint Service requires the input of a latitude and longitude and generates a map of the locality of that given point. The Translation Service transforms a specific instance of an Address into an instance of Location. With the new augmented domain ontology, shown in Figure 5, our algorithm would discover an alternative workflow. This new workflow flows from Business to Address to the Translation Service to Location to MapPointService to Map. If the concept Location was populated with instance data for our given input instance of business (i.e., the instance of Business with the Name "The MITRE Corporation"), then our algorithm would discover a third potential workflow. This workflow goes from Business to Location to MapPointService to Map.

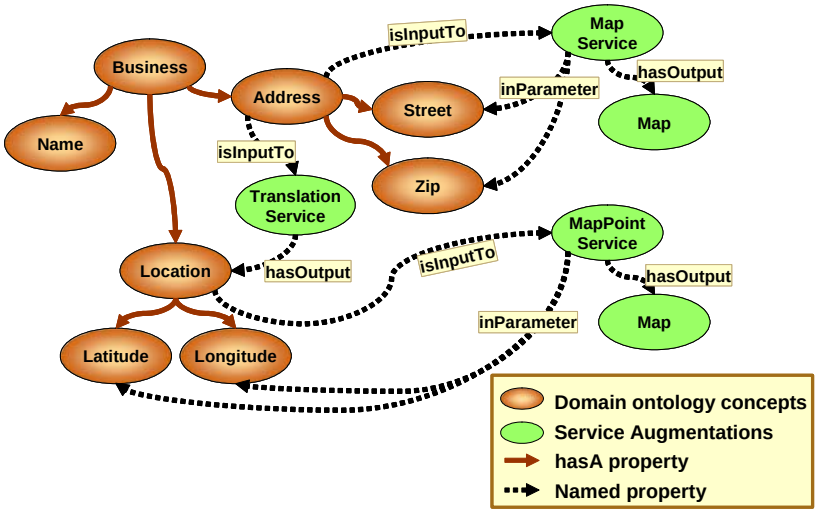

Figure 5. New Service Adds Additional Workflow

Adding more Web services demonstrates the extensibility of this approach and the discovery of additional workflows. In our implementation, when more than one workflow is discovered the workflow options are presented to the user who is asked to select a desired workflow. User selection of a desired workflow results in invocation of the Web Service Generator.

Web Service Generation. The Web Service Generator takes the user selected workflow and automatically generates the software needed to execute the given workflow. The Web Service Generator uses information contained in the Web Service ontology to chain together the actual software methods and necessary parameters to create this "glue-code". The Web Service Generator persists this automatically generated code and then executes it, resulting in presentation of the desired output to the user.

The Family – Extending and Generalizing the Approach

We used a simple example to show that one can use ontologies to marry domain semantics and Web services. This simple example demonstrated that mapping Web services to a domain ontology can result in several automatically discovered potential workflows that meet a user's needs. Additional workflows could be discovered not only through the addition of more Web services, but also through the addition of new domain concepts or ontologies.

One concern that naturally arises is the potential for an explosion in the number of possible workflows. Keeping the number of discovered workflows to a manageable size could be handled in a number of ways. For example, one could use reasoning to control what workflows are presented as options to the user. One example of using reasoning would be to use rules to halt the workflow discovery process after some user-specified number of workflows were discovered. Another example of using reasoning would be to use rules to suppress certain paths in the augmented domain ontology. For instance, one could suppress certain paths by specifying that a particular property or concept in the augmented domain ontology may not be used in the workflow discovery process.

The workflow discovery process shows that a software application can discover Web service workflows. Our Web service generator demonstrates that a software application can use information contained in the workflow, along with information on the invocation of Web services contained in an ontology, to automatically generate the "glue-code" needed to execute the workflow. This glue-code combines the code needed to invoke each Web service and to chain these invocations together to achieve a desired result. This eliminates the need for the programmer to find Web services that provide the desired result, determine their necessary input parameters, write code to properly format the input parameters, and write code to sequence the invocations of multiple Web services together into a desired workflow. Although our approach requires that Web services be mapped to relevant domain ontologies, once this is accomplished these Web services can be composed into any number of Web service workflows. Hopefully, even the simple example we used shows the power of using such a flexible and extensible Web service orchestration approach.

SUMMARY

In this paper, we showed that by mapping Web services into a domain ontology we can automatically discover Web service workflows that satisfy a user request. We also showed that mapping additional Web services can result in new workflows

being discovered. Further, we demonstrated how instance data resident in databases can be mapped to the domain ontology and used as input to Web services in the process of satisfying a request. We are currently extending our approach to work with multiple domain ontologies.

REFERENCES

[1] http://www.uddi.org/specification.html
[2] http://www.w3.org/2002/ws/desc/
[3] http://www.w3.org/2001/sw/WebOnt/
[4] OWL Web Ontology Language Overview, W3C Proposed Recommendation, 15 December 2003. Available online at: http://www.w3.org/TR/2004/REC-webont-req-20040210/
[5] http://www.daml.org/services/
[6] http://www.daml.org/
[7] http://www.landcglobal.com/
[8] Network Inference Construct and Cerebra Server. Details available at: http://www.networkinference.com/