Adding Semantics to Web Services Standards

Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, John Miller
Large Scale Distributed Information Systems (LSDIS) Lab
Department of Computer Science, University of Georgia
Athens, GA 30602

Abstract

With the increasing growth in popularity of Web services, discovery of relevant Web services becomes a significant challenge. One approach is to develop semantic Web services where by the Web Services are annotated based on shared ontologies, and use these annotations for semantics-based discovery of relevant Web Services. We discuss one such approach that involves adding semantics to WSDL using DAML+OIL ontologies. Our approach also uses UDDI to store these semantic annotations and search for Web services based on them. We compare our approach with another initiative to add semantics to support Web service discovery, and show that our approach may fit current standards-based industry approach better.

Keywords: Web services, Semantic discovery, Semantic Web, Ontologies

1. Introduction

“Web services are modular, self-describing, self-contained applications that are accessible over the Internet” [Curbera et al, 2001]. They have been identified as the technology for business process execution and application integration. Given, the dynamic environment in e-businesses, the power of being able to find Web services on the fly to create business processes is highly desirable. A key step in achieving this capability is the automated discovery of Web services. Currently, the industry standards for Web services are Web Services Description Language [WSDL, 2003] and Universal Description Discovery and Integration [UDDI, 2000] specifications. Web services are described using WSDL definitions and advertised in UDDI registries. The current discovery mechanism supported by UDDI is not powerful enough for automated discovery. The main inhibitor is the lack of semantics in the discovery process and the fact that UDDI does not use information in the service descriptions during discovery. This makes UDDI less effective, even though it provides a very powerful interface for keyword and taxonomy based searching. The key to semantic discovery of Web services is having semantics in the description itself (i.e., “formally self described” [Sheth and Meersman, 2002]) and then using semantic matching algorithms to find the required services.

Ontologies [Gruber, 1993] have been identified as the basis for semantic annotation that can be used for discovery. Ontologies are the basis for shared conceptualization of a domain, and comprise of concepts with their relationships and properties. Use of ontologies to provide underpinning for information sharing and semantic interoperability has been long realized [Gruber, 1991; Kashyap and Sheth, 1994; Wache et al, 2001]. By mapping concepts in a Web resource (whether data or Web service) to ontological concepts, users can explicitly define the semantics of that resource in that domain. An approach for semantic Web service discovery is to have the ability to construct queries using ontological concepts in a domain. This in turn requires mapping concepts in Web service descriptions to ontological concepts. By having both the description and query explicitly declare their semantics, the results will be more relevant than keyword matching based information retrieval.

There have been a number of efforts to add semantics to the discovery process. An early work in this area has been the creation of DAML-S [Ankolenkar et al, 2002], which uses a DAML+OIL based ontology for describing Web services. While DAML-S provides the expressiveness required for automated discovery, it does not obviate the need for WSDL to represent communication level details of Web services. The recently released version [DAML-S 0.7, 2002] of DAML-S suggests using WSDL in conjunction with DAML-S for Web service descriptions. In this paper, we explore the possibilities of adding semantics to WSDL and UDDI to achieve sufficient expressiveness to automate the discovery process. Our approach involves relating concepts in WSDL to DAML+OIL ontologies in Web service description and then providing an interface to UDDI that allows querying based on ontological concepts. WSDL has been accepted as the industry standard for Web service description. If extending it without adding significant complexity could provide the same functionality as DAML-S, our approach may be
preferable than migrating from WSDL to DAML-S which entails the use of a more complex and non-standard framework. Since our approach is backward compatible with existing WSDL standards, service providers also have an option to describe and publish their services with or without semantics. We further provide a matching algorithm to use this semantic information for Web service discovery that considers not only inputs and outputs, but also functional specification of operations and effects. Matchmaking with DAML-S as described in [Paolucci et al, 2002b] does not consider operations. In this paper we first outline our approach for adding semantics in WSDL and UDDI. Then, we discuss our semantic discovery algorithm. Thereafter, we compare our approach with DAML-S for adding semantics to Web services and using it for discovery. The rest of the paper is organized as follows. Section 2 outlines the related work. Our approach of adding semantics to WSDL and UDDI are discussed in Sections 3 and 4, respectively. Semantic Web service discovery is discussed in Section 5. Section 6 provides a comparison of our approach with DAML-S approach. Finally in Section 7, we present conclusions and future work.

2. Adding Semantics in WSDL

Currently Web services are described using WSDL descriptions, which provide operational information. Although WSDL descriptions do not provide (or at least explicate) semantics, they do specify the structure of message components using XML schema constructs. We suggest adding semantics to WSDL using extensibility in elements and attributes supported by WSDL specification version1.2. These extensions are similar to the extensions suggested for ServiceGrounding in DAML-S. Using extensibility as explained in following sections as an alternative for DAML-S provides almost equivalent expressiveness.

2.1. Mapping Operations to Ontological Concepts

Recent tools like Web Services Invocation Framework [Dufll, 2001] allow invoking Web services if the locations of the WSDL file and the name of the operation are known. So service discovery involves not only locating the WSDL description, but also the relevant operation to invoke. Each WSDL description may have a number of operations with different functionalities. The WSDL file shown in figure 1 represents a sample Web service and has operations for both booking and canceling flight tickets. In order to add semantics and to find relevant operations, these operations should be mapped to concepts in appropriate DAML+OIL ontologies which depict functionality of operations. In figure 1, the operations buyTicket and cancelTicket are mapped to ontological concepts BookTicket and CancelTicket, using the attribute operation-concept respectively. This allows users to search for operations based on ontological concepts. An approach to store the mapping between operation names and ontological concepts in UDDI is discussed in section 4.

2.2. Mapping Message Parts to Ontological Concepts

Message parts, which are input and output parameters of operations, are defined in WSDL files using XML schema constructs. XML schemas could be used as shared definitions of concepts. Since service providers typically embed the schema definitions as inline elements along with service descriptions, it becomes difficult to share and reuse them. Ontologies, which are more expressive and meant for sharing definitions, can be used to annotate the message parts in WSDL. Using ontologies not only brings user requirements and service advertisements to common conceptual space, but also helps to apply reasoning mechanism to find a better match. Hence by using DAML+OIL ontologies in WSDL, the semantics implied by these structures in service descriptions, which are known only to the writer of the description (provider of Web service), can be made explicit. In Figure 1, the WSDL constructs input TravelDetails and output Confirmation are mapped to ontological concepts TicketInformation and ConfirmationMessage, respectively.

2.3. Adding New Tags for Preconditions and Effects

Each operation may have a number of preconditions and effects. The preconditions may be some logical conditions which must be true for executing the operation. Effects are changes in the world after the execution of the operation. We propose adding precondition and effect elements as children of the operation element in WSDL. Figure 1 shows the added preconditions and effects to each of the operations in the WSDL description. The operation buyTicket has the

---

1 The names spaces LSDISOnt and LSDISExtension respectively contain TravelServices ontology and the extended WSDL schema used in the examples

2 In a related work, we are investigating semantic heterogeneity between schema constructs in WSDL and ontological concepts during mapping of message parts to ontology [Patil et al, 2003]
precondition and effect mapped to ontological³ concepts ValidCreditCard and CardCharged-TicketBookedReadyForCollection respectively.

We believe that preconditions are important for invoking a service and that effects are the important for Web service selection. After matching services based on operations, inputs and outputs, effects could be used to select the most relevant service. It is possible for a number of operations to have the same functionality, as well as, the same inputs and outputs, but different effects. For example, there could be an operation called bookTicketAndSend, with the same functionality, inputs and outputs as buyTicket in Figure 1, but with a different effect called “CardCharged-TicketBookedAndSent”. Upon execution, bookTicketAndSend sends tickets to the user of the service rather than making them ready for collection. In this case, depending on the requirements of the user, the most relevant operation can be chosen.

3. Adding Semantics in UDDI

We provide semantic discovery using UDDI by doing the following two tasks. Firstly, we store the semantic information about Web services mentioned in section 3 in the existing structures of UDDI. Secondly, we provide an interface to construct queries that use that semantic information. This approach is similar to the one suggested by [Paolucci et al, 2002a], which maps DAML-S to UDDI structures, but is consistent with the use of industry standard WSDL rather than requiring DAML-S.

UDDI only supports a limited form of semantics using tModels, which are used to characterize and categorize businesses and their services. During Web service publication, ontological concepts representing operations and their message parts of the WSDL descriptions of the Web service are stored using the UDDI structures, tModels and CategoryBags. tModels are metadata constructs in UDDI data structure that provide the ability to describe compliance with a specification, a concept or a shared understanding. They have various uses in UDDI registry. Commonly agreed specifications or taxonomies can be registered with UDDI as tModels. They can also be used to associate entities with individual nodes in taxonomies. When a tModel is registered with UDDI registry, it is assigned a unique key, which can be used by entities to refer to it. To categorize entities in UDDI, tModels are used in relation with CategoryBags, which are data structures that allow entities to be categorized according to one or more tModels. Using the new grouping construct keyedReferenceGroups in UDDI version 3 specifications, categorization using tModels can be grouped. We propose using the keyedReferenceGroup, along with tModels to group operations with their inputs and outputs.

To represent the semantic information in UDDI, we have created four tModels in that registry. The first tModel represents the ontology of concepts representing functionality of operations in a relevant domain, the second and third represent the ontologies of input and output concepts respectively. Finally, the fourth tModel represents the grouping of each operation with its inputs and outputs. These tModels are linked with the respective ontologies using overviewURL tag of these tModels. All the tModels could as well be linked to a single comprehensive ontology. As shown in Figure 2, two keyedReferenceGroups have been created to represent two operations, BookTicket and CancelTicket along with their inputs and outputs. Each keyed reference has a keyValue which represents an ontological concept and a tModelKey which represents the ontology itself. For example, the tModel OPERATION_CONCEPTS is used to store the mapping between an operation and a concept in ontology. It contains the name of the operation as keyName and the ontological concept it is mapped to as keyValue. Similarly the inputs and outputs of each operation are mapped using INPUT_CONCEPTS and OUTPUT_CONCEPTS tModels, respectively. Each operation along with its inputs and outputs is grouped using MAPPINGGROUP tModel into keyedReferenceGroups.

4. Semantic Web Service Discovery

Semantic annotations added in WSDL and in UDDI are aimed at improving discovery and composition of services. In this section we briefly describe our mechanism for template based ontology enabled discovery. Figure 3 shows the conceptual process of mapping WSDL constructs given in Figure 1, to the nodes in a domain specific ontology. This mapping is then stored in UDDI during Web service publication. As shown in the figure, the operations buyTicket and cancelTicket are mapped to the nodes BookTicket and CancelTicket, respectively, the input concept of WSDL file TravelDetails and output concept Confirmation are mapped to the TicketInformation node and ConfirmationMessage in the TravelServices ontology, respectively.

We have developed a three phase algorithm for semantic Web service discovery which requires the users to enter Web service requirements as templates constructed using

³ A discussion of creating an ontology depicting preconditions and effects are beyond the scope of this paper.
ontological concepts. In the first phase, the algorithm matches Web services based on the functionality they provide. In the second phase, the result set from the first phase is ranked on the basis of semantic similarity [Cardoso and Sheth, 2002] between the input and output concepts of the selected operations and the input and output concepts of the template respectively. The optional third phase involves ranking based on the effects. Figure 3 shows the creation of a template using ontological nodes for semantic discovery of services. The template has the operation concept BookTicket, the input concept TicketInformation and the output concept ConfirmationMessage. The template created by the user is converted to a UDDI query by our interface [Verma et al, 2003]. This template would map to an UDDI query which first searches for all Web services categorized using a keyedReferenceGroup which has the BookTicket mapped to the operation tModel. The result set is then ranked based on the semantic similarity [Cardoso and Sheth, 2002] between the input concepts of the returned Web services to the input concepts (TicketInformation) of the template and the output concepts of the returned Web services to the output concepts (ConfirmationMessage) of the template.

5. Related Work

DAML-S [Ankolenkar et al, 2002] is based on DAML+OIL and it provides an ontology markup language expressive enough to semantically represent capabilities and properties of Web services. Its goals are to achieve automatic Web service discovery, invocation, composition and execution monitoring. DAML-S has an upper ontology which characterizes Web services using three types of knowledge about the services - ServiceProfile, ServiceModel and ServiceGrounding. ServiceProfile is used to describe what a Web service does, ServiceModel describes how it works and ServiceGrounding is used to specify how to access it. [Paolucci et al, 2002b] presents a mapping engine to match service advertisements with requests. It provides a semantic algorithm to match inputs and outputs of Web service requests with inputs and outputs of Web service advertisements during the matchmaking process. It adds an additional mapping layer over UDDI and uses DAML-S as service description language to provide better service discovery than keyword based search.

The latest version (version 0.7) of DAML-S suggests using a WSDL file along with a DAML-S description to represent a service. Our approach involves annotating and extending WSDL constructs with DAML+OIL ontological concepts. Since ServiceProfile is used by DAML-S to describe and discover a Web service, all our extensions aim to provide the same functionality as ServiceProfile for Web service discovery. Some of the details in the ServiceProfile like service provider details are already supported by UDDI, so we have not added them to WSDL. We have used an approach similar to [Paolucci et al, 2002a] to store the semantic information about inputs, outputs and operations of a WSDL description in UDDI. As argued earlier, our approach has the advantage of an ontology-based approach that fits better with existing industry norms and standards, rather than requiring new infrastructure as needed by DAML-S. While the matching algorithm provided in [Paolucci et al, 2002b], uses only inputs and outputs to search for required Web services, our discovery algorithm first selects the services using ontological concepts representing functionality of operations, and then uses inputs and outputs to prune the search. In the example given in figure 1, both the operations BookTicket and CancelTicket have the same inputs and outputs, but they have different functionalities, therefore, searching just based on inputs and outputs would lead to incorrect results. Our algorithm also recommends using other details like preconditions and effects from the WSDL file to ensure that the operation matches exact requirements.

[Ogbuji, 2000] discusses representing WSDL in RDF instead of XML. Our approach uses DAML+OIL ontologies in RDFS format to add semantics to Web service descriptions. The WSMF architecture [Bussler et al, 2001] discusses using semantics at different levels of Web services stack. It proposes a conceptual framework that provides a model to describe Web services and their composition. Their approach is not specific to any of the industry standards as they aim to use mediation to adapt to any standard.

A significant amount of recent research has focused on effective discovery of services which is the key required capability of the Web services framework. The discovery mechanisms suggested to improve keyword based discovery range from categorization and domain independent characterization of services [Dumas et al, 2001] to better techniques exploiting semantic representations of the services. [Trastour et al, 2001] analyses the problem of matchmaking and highlights the need for metadata for better results and suggests requirements for advanced matchmaking as high degree

---

4 Functionality is specified using ontological concepts that map to operations
5 Preconditions and effects are not shown as they are optional
6 Our implementation uses UDDI Version 1 API. Hence we have grouped operations with its inputs and outputs using the keyName parameter instead of keyedReferenceGroup
of flexibility and expressiveness; ability to express semi-structured data; support for type and subsumption; ability to express constraints over ranges of possible values as well as definite values of a specification. They have also observed that UDDI does not allow much expressiveness and flexibility. With our approach we intend to DAML+OIL ontologies to add these capabilities to UDDI data structure. Our approach involves storing semantically annotated Web service descriptions in UDDI, and using that semantic information for querying. [Bernstein and Klein, 2002] suggests using process models to capture service behavior. The process models are created by indexing services to nodes in the process ontology. They also provide a process querying language for logic based querying of the process ontology to retrieve the most relevant services. Our approach maps operations in Web service descriptions to ontological concepts that represent functionalities and querying is based on templates.

6. Conclusion and Future Work

In this paper, we have presented our approach of adding semantics to Web services. The current approach involves using DAML-S for adding semantics to Web services description. Since WSDL and UDDI are current industry standards, we believe that a pragmatic solution for adding semantics to Web services would be to add semantics to both of them, instead of creating a new language. While DAML-S is a highly expressive language, in which the features are meant not only for discovery but also for composition, execution and monitoring. In this paper, we have concentrated on adding enough semantics to WSDL using DAML+OIL ontologies, for it to have the same descriptive power as DAML-S for discovery. We have also discussed an algorithm for semantic discovery of Web services which uses functionality of the service as the main criterion for search. The main contributions of this work are

- Using the extensibility feature of WSDL to add semantics to service descriptions
- Using UDDI data structures to represent grouping of operations with their inputs and outputs

We are currently working on enhancing WSDL to make them better suited for service selection in e-commerce. Some of the features we intend to add are functional and behavioral attributes like QoS and constraints. We are also working on developing richer ontologies to depict functionality of operations [Bernstein and Klein, 2002], preconditions and effects. To fully utilize the potential of DAML+OIL ontologies which we use for adding semantics to WSDL and UDDI, we are working on developing a more powerful logic based querying mechanism.

7. References


Figure 1: WSDL File Extended with Semantic Constructs

7 The extended elements and attributes have been underlined and are optional.
Figure 2: Representation of Operations, Inputs and Outputs in UDDI

Figure 3: Semantic Annotation, Publication and Discovery

---

For simplicity of depicting, TravelService Ontology is shown in figure 3 with separate classes called data and operations, meaning TicketInfo or ConfirmationMessage are of type data, BookTicket or CancelTicket are of type Operations, in TravelServiceOntology.