Semantics Based Web Services Discovery

Shou-jian Yu
College of Computer Science and Technology, University of Donghua, Shanghai, China

Jing-zhou Zhang
Shanghai Development Center of Computer Software Technology, Shanghai, China

Xiao-kun Ge
Shanghai Development Center of Computer Software Technology, Shanghai, China

Guo-wen Wu
College of Computer Science and Technology, University of Donghua, Shanghai, China

Abstract - *Web services have become a new wave of Internet technology development. It is a new solution for dynamic business interactions over the Internet. A critical step in the process of reusing existing Web services is the discovery of potentially relevant components. Current Web services technology only provides syntactic description for Web service. In this paper, we first analyze the limitations for current Web service standards and point out that semantic description is the basis for automatic service discovery. We put forward a new semantics based Web services discovery framework by extending UDDI framework. This framework provides operation level service discovery compared to UDDI architecture. The discovery process is in two stages. If exact match is not met, the match engine will search for suitable services to compose the target request. The composition algorithm is also analyzed. The method of semantic Web service annotation is discussed. We use an ontology based approach to capture real world knowledge for semantic service annotation. The prototype we have implemented gives a whole understanding of the discovery model we have proposed.*

Keywords: Web service discovery, Semantic, Annotation, UDDI

1 Introduction

Faced with decreasing time-to-market and increasing requirement volatility, software development processes are increasingly relying on reuse of existing software. Web services are modular and self-describing applications that can be mixed and matched with other Web services to create new software components [1].

The Web services set of standards is aimed at facilitating and improving the quality of component-based applications on the web. It consists of a set of related specifications, defining how reusable components should be specified (through the Web-Service Description Language – WSDL [2]), how they should be advertised so that they can be discovered and reused (through the Universal Description, Discovery, and Integration API – UDDI [3]), and how they should be invoked at run time (through the Simple Object Access Protocol API – SOAP [4]).

The lack of semantics in description creates inefficiencies in exploiting the Web service discovery. Describing Web service with semantics provides the ability for automatic Web service discovery, invocation, composition and interoperation, and Web service execution monitoring. Current standards focus on syntactic description of Web services. To overcome such limitations, the semantic web technology must be applied to Web service. Recently there is an important initiative in this respect, namely, DAML-S [5], which is OWL-S now, the newest version [6]. It is a comprehensive effort based on OWL [7] defining an upper ontology for Web service description.

In this paper, we develop a framework for semantically Web service discovery where we incorporate the semantics and integrate it with UDDI registries. Our aim is to ground the discovery of Web services on a semantic comparison between a client query and available Web services. We first analyze the description requirements for Web service discovery and limitations for current standards in section 2. Section 3 gives the detail discussion of all the components of the framework.

The contribution of this paper lies in three fold. First, we conclude the limitations of WSDL and UDDI for automatic discovery. These standards concentrate on syntactic description for Web service. Second, on the basis of analysis of semantics lack for current standards, we put forward the semantics based Web services discovery framework by extending UDDI framework. This architecture supports both service publishing and service discovery. Compared to previous work, the discovery process includes two steps. First is the direct discovery by exact matching. If this step is not successful, the composer looks for several services to compose for the request. Third, We use an ontology based approach to capture real world knowledge for semantic service annotation. We model a

This work has been partially supported by the Key Programs of Science and Technology Commission Foundation of Shanghai, China under contract 05DZ11C06, and the project: Visual Document Management System for Compoment Development.

concept from two aspects: behaviors and attributes. Behaviors are mapped for annotations of operations in Web service. Attributes are mapped for annotations of i/o in operations. This supports the finest granularity annotation for Web service. In the end, we implement a simple prototype to demonstrate our work.

2 Problem Description

Finding and matching of Web services is fundamentally semantic in nature [8]. The current industry standards can describe the interface of services and how the services are deployed well (via SOAP and WSDL), but are limited in their ability to express what the capabilities of the services are. This lack of semantics is the result of the current syntax-oriented interface representations that cannot express the context in which the services operate and the relationships among various entities in that context.

2.1 Description Requirements For Web Service Discovery

The description of Web service capabilities is essential for classifying, discovering and using a service. It needs to be understandable by humans as well as by machines [9, 10]. This means that each service attribute must be described at both syntactic and semantic level. Syntactic information is concerned with the implementation aspects of a service and thus tailored towards the programmers' requirements. Semantic information is concerned with the conceptual aspects of a service aiming at facilitating end-users by shielding off the lower level technical details, as well as to facilitate developers to find services that best match their needs and to enable automatic service discovery [11].

Let us consider a stock quote service, which takes as input a string denoting the stock symbol and returns the stock quote as a number. The syntactic information denotes that the input parameter is a string and the output is a number, whereas semantic information conveys the real world meaning of the string and the number in the context of stock quote markets. Depending on whether the service requestor is an end-user, a developer or a machine, different kinds of service description are required. For the end-user, only semantic description is needed whereas developers or machines need both semantic and syntactic information.

2.2 Limitations for WSDL and UDDI

WSDL is an XML grammar for specifying properties of a Web service such as what it does, where it is located and how it is invoked, i.e., it describes only the functional and syntactic aspects of a service. WSDL does not support non-functional information of services. For example, it is not possible to indicate the geographic region that a weather service is provided for or the charge associated with the service.

UDDI is an industry effort to provide directory services for Web services offered by businesses. It allows businesses to publish their services in a directory and enable other business entities to locate partners and to form business relationships based on the Web services they provide. UDDI provides a set of search facilities for finding businesses, and their services. Services can be searched by specifying business name, service name, service category or Tmodels. However, UDDI in its current form is limited in its search services by its inability to extend beyond the keyword-based matches.

First, UDDI does not capture the relationships between entities in its directory and therefore is not capable of making use of the semantic information to infer relationships during search. For example, a rental car service might advertise itself under 'Car Rental Services' in UNSPSC category but a request that is looking for car rental services under 'Passenger Transport' category would not find any matches although 'Car Rental Services' is a sub category under 'Road Transport', which in turn is a sub category of 'Passenger Transport'.

Second, UDDI supports search based only on the high-level information specified about businesses and services. It does not get to the specifics of the capabilities of services during matching. For example, UDDI can search for services that offer car rental services such as creating a reservation, updating a reservation, getting rental status etc. However, it cannot search for a service that can create a reservation by taking information such as user name, credit card information, rental pick up location, rental drop off location and drivers license and returning a reservation number.

Third, the search facilities in UDDI support only direct matches. In cases where no direct matches are available but a set of services can be composed to fulfill a request, UDDI fails to provide any search results because it does not look beyond direct matches.

3 Semantics Based Web Services Discovery Framework

Current Web service standards focus on technical conventions which allow parties to exchange information in a standardized manner. They solve many problems on the technical level but the semantics of Web services and Web service descriptions as a whole are not addressed by them. In this section, we put forward a discovery framework incorporating semantic description by extending current UDDI architecture. This architecture supports publishing and discovering of services. Fig. 1 shows a modular architecture of our semantically enhanced

UDDI directory. We use domain ontology to provide an understanding of the static domain knowledge that facilitates knowledge sharing and reuse [12]. This is the basis for semantic description. For example, a travel ontology can codify the relationships such as a compact car 'is a' car which 'is a' vehicle.

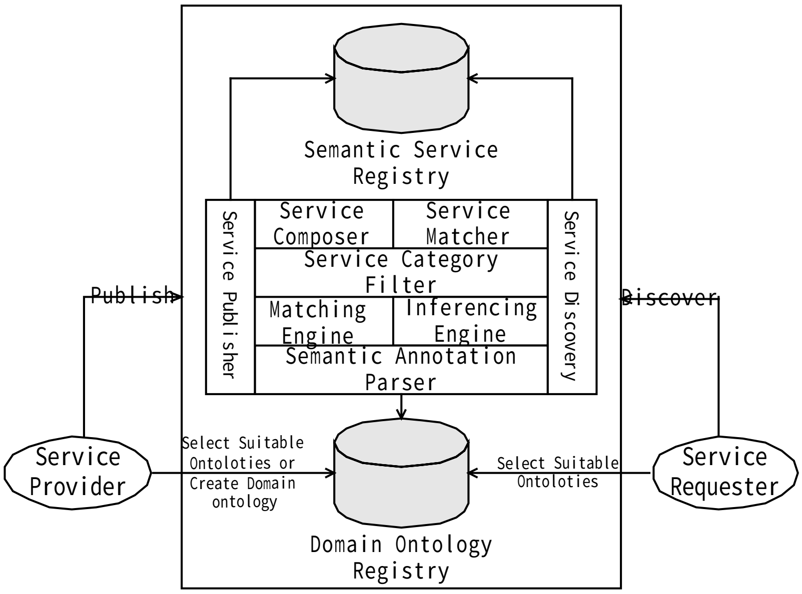

Fig. 1. Semantics based Web service discovery framework

3.1 Service Publishing

First, service providers describe the terms and concepts in their problem domain and their interrelationships to establish the ontology for describing the capabilities of their services. This is done by either creating an ontology document or selecting a suitable ontology from an existing ontology registry. This ontology is a document or a file that formally defines relations among terms. For example, if a rental car agency wants to publish its car rental services in UDDI registry, it would first describe the car rental domain in an ontology with domain classes such as reservation, pickup location, drop-off location, user, confirmation number, credit information, business affiliation, reservation start date, and duration. In the next section, we will give a detail discussion of the semantic relationship for service description.

Next, service providers annotate their services with semantic information. This contains information about the service provider, the functional attributes of a service (such as quality rating, quality guarantee, geographical radius, etc.), and the properties of the service namely the inputs, outputs, preconditions and effects. All of these together describe what a service is capable of doing. The properties of service semantics refer to the concepts defined in the domain ontology to express the context. After annotating services with semantics, a service provider publishes them in a UDDI registry. The publisher module uses this information to publish the given services under the specified UDDI taxonomy.

3.2 Two-steps Service Discovery

The service discovery module receives requests and executes the inquiry. First, the service category filter is applied to the service request. It performs a UDDI category-based search to retrieve all those services that fall under the specified set of categories in given taxonomies. This is performed using the standard UDDI find method. These filtered set of services are then passed into semantic matching engine. In essence, the matching engine matches the inputs, and outputs of a service request with those of a service. Two properties are considered a match if they either match exactly, or as defined by some relationship that can be inferenced from the ontology using an inferencing engine.

The matching engine first looks for any services that directly match the given request. This is the first step for discovery. A match is considered a direct match if a single service meets the requirements of a request exactly. If no direct matches can be found, our semantic matching engine automatically finds ways in which two or more services could be composed to meet the original request. This is the second step using service composition techniques to meet the service requestor. The next section will discuss the combination of several services for service discovery.

3.3 Web Service Composition

A service is composed of several operations. The input/output of each operation tells us what type of document needs be provided in order to execute it, as well as the types of documents that will be returned upon successful and unsuccessful execution. It also gives us information on possible compositions of services. For example, if a particular service has an operation "buyBook", which takes as input an "isdnCode", and another operation "getISDN" (from a different service) outputs values of the same type, then we know that the latter operation is a composition of the former operation. This process can be implemented automatically. Sometimes a goal needs to be composed recursively. The detailed algorithm is as follows.

In order to express the service requestor' need, we use a set of output to express the goal, i.e. $S(O_1,O_2,\ldots O_n)$. The service requestor may provide some local information. For example, someone may want to query the temperature of Beijing in China, the location, i.e. Beijing, is an input for the objective Web service. Thus we express this kind of local information as a set of input, i.e. $S(I_1,I_2,\ldots I_m)$. The algorithm executes as follows.

The algorithm takes $S(O_1,O_2,\ldots O_n)$ as the goal to be achieved and searches operations whose outputs are of sufficient similarity to the goal, inferencing from the domain ontology registry. The operation with the most similar output is selected first. If there are more than one operations (e.g. the same service is provided by two different companies), the algorithm will select one of them with the least number of inputs in $S(I_1,I_2,\ldots I_m)$.

If the total outputs can't be included in any operation, the operation with the maximum number of the total operations is selected and the left outputs are used as the goal for searching for operations with the same algorithm.

If $S(I_1,I_2,\ldots I_m)$ can satisfy these services found in the above procedure, i.e. their inputs are included in $S(I_1,I_2,\ldots I_m)$, the algorithm terminates and the goal is accomplished by the composition of these Web services. Otherwise, the algorithm takes inputs or the left inputs excluded from $S(I_1,I_2,\ldots I_m)$ of these Web service as the goal to searches for operations and repeats the above procedure until $S(I_1,I_2,\ldots I_m)$ can satisfy these found services or a search limit is exceeded, i.e. the algorithm calls itself recursively. Then the request is composed by all these Web services.

4 Web Service Semantic Annotation

Current Web services standards have focused on operational and syntactic details for implementation and execution of Web services. This limits the search mechanism for Web services to keyword based searches. Research in the Semantic Web area has shown that annotation with metadata can help us solve the problem of inefficient keyword based searches. Adding semantic information to syntactical Web service definitions can help interpret the purpose and usage of Web service. This is on the premise that a Web service references to a proper ontology which provides a computer-interpretable description of the service. The ontology relates the domain concepts together for the whole semantics. A Web service can be expressed as a set of operations. Each operation implements one functionality. An operation is specified by its name, its input and output message types, i.e. o: =<name, t_{in}, t_{out}>. Web services discovery is in fact the matching of inputs/outputs of operations in Web services. While searching for Web services, relevant domain specific ontologies can be referred to, thus enabling semantic matching of services. In this section, we discuss the ontology based Web service annotation method.

4.1 Organizing Concepts into Ontology

An ontology is a shared formalization of a conceptualization of a domain [13]. In the Semantic Web, ontologies are used to assign commonly agreed upon semantics or interpretation to particular concepts. In the following, we describe what role ontologies play in the Web services description and how they can be used to give Web service comprehensive semantic annotation.

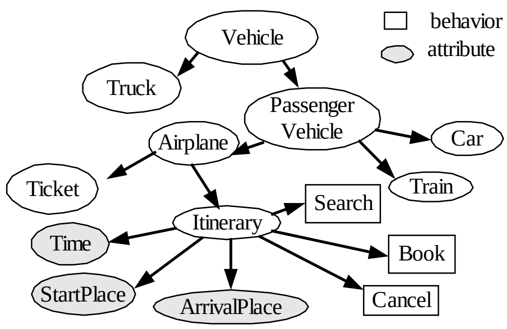

Fig. 2. Organize concepts into ontology

In our approach, we identify two aspects of semantics for Web service description elicited from object-oriented model: an object can be real entity with practical meaning or an abstract entity with no real meaning. A class is the common characteristics abstraction of objects. The class has attributes and behaviors. From this point, we also model concept into domain ontology from both attributes and behaviors aspects. As shown in Fig. 2, the concept *itinerary* has attributes as *startPlace, arrivalPlace and Time* etc. It is also associated with behaviors as *search, book, cancel* etc. The concept describes what functionality a Web service aims at. As a Web service is implemented by several functions, i.e. operations, the behaviors aspect is used to describe operations of a Web service. As an operation has inputs and outputs, the attributes aspects can be used to describe the i/o interface of operations. From the left part of Fig. 3, we can see the simplified structure of WSDL. By this method, Web service is semantically described in different granularity, which facilitates different service level discovery.

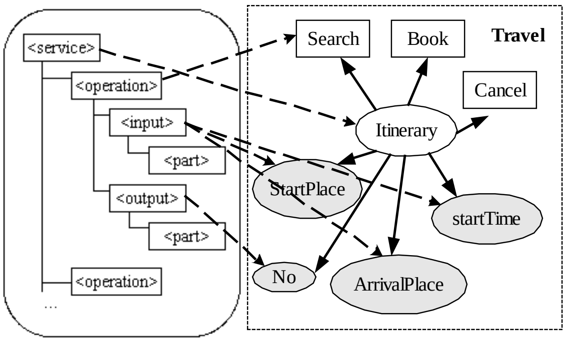

Fig. 3. Incorporating semantics into WSDL

We attempt to use minimum information to give Web service comprehensive annotations although ontology can capture much information of real world knowledge and domain theory. The detailed information can be incorporated into WSDL or UDDI. The next section will discuss how to incorporate semantics into WSDL.

4.2 Incorporating Semantics into WSDL

WSDL is the current standard of the description of Web service. The syntax of WSDL is defined in XML Schema. A WSDL document describes the location of a Web service, its available operations and their associated messages and data types as well as the format of their result values. The XML Schema types of message parameters may be defined using a <types> element. A <message> element is needed to compose such data types into messages. Messages need to be grouped into operations, which may define an <input>, an <output> and a <fault> message. We add semantics to Web services by mapping service, operation, input and output in their descriptions to concepts in the domain specific ontologies. The structure is shown as Fig. 3. The semantics annotated WSDL document is exemplified in the following example code.

```
<types>
    <xsd:element name="StartCity" Ont-
Concept="Itinerary:startPlace"
            minOccurs="1" maxOccurs="1"
type="xsd:string">
    <xsd:element name="ArrivalCity" Ont-
Concept="Itinerary:arrivalPlace"
            minOccurs="1" maxOccurs="1"
type="xsd:string">
    <xsd:element name=" time" Ont-
Concept="Itinerary:startTime"
            minOccurs="1" maxOccurs="1"
type="xsd:string">
……
</types>
……
<portType>
    <operation name="SearchItineary" Ont-
Concept="Itinerary:Search">
        ……
    </operation>
</portType>
    ……
<service name="ItineraryService" Ont-
Concept="Travel:Itinerary">
    ……
</service>
```

As the above WSDL code shows, the Web service is annotated in three aspects: service *ItineraryService* is annotated by *Travel:Itinerary*, operation *SearchItineary* is annotated by *Itinerary:Search* and i/o element *StartCity* is annotated by *Itinerary:startPlace* etc.

5 Implementation

In this section, we provide an initial prototype to illustrate the composition model we proposed. The WSDL parsing tool is in progress. We use relational database to simulate elements of the WSDL document. We design three tables as follows.

Element (*Name*, *Operation*, *i/oType*, *Description*) is used for simulating global data schema. An element is associated with one or several operations and an expressive description. *Webservice* (*ServiceName*, *Operation*) is used for representing the operations for a service. *Semantics* (*Name*, *RelatedElement*, *SemanticType*) is used for storage of semantic relationship between elements.

The prototype interface is demonstrated in Fig. 4. The composer should provide publisher with the global schema. In this prototype, user can query by web service (name), element or description. In the right tree view, it provides a level structure for representation. The first level is web service, and the second is the associated operations, then the related inputs/outputs parameters and their semantic related elements. User can get a whole idea from the view.

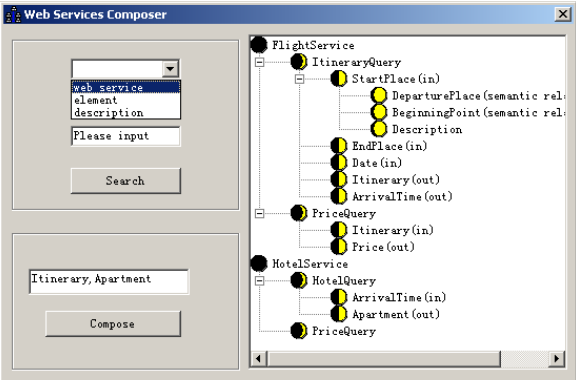

Fig. 4. Web services discovery prototype

User can input composition goal parameters in the left-corner textbox. For example, we take the scenario in a travel agency as an example [14]. User inputs *Itinerary* and *Apartment* as the goal for flight reservation and hotel reservation. The prototype generates two web services as shown in the tree view. One is *FlightService*, and the other is *HotelService*. This prototype also provides the ability of modifying the model. In the prototype, user can modify every node with a mouse-click on it, add or drop any node, drag a node to associate it with other node, etc.

6 Conclusions and Future Work

Location of Web services is inherently a semantic problem, because it has to abstract from the superficial differences between representations of the services provided, and the services requested to recognize semantic similarities between the two. Current Web services technology based on UDDI and WSDL does not make any use of semantic information and therefore fails to address the problem of matching between capabilities of services. They provide limited support in automating the Web service delivery tasks. Mainly, the lack of a machine processable language that enables to make explicit the semantics of service descriptions limits the usability of Web services.

In our work, we present a new framework which incorporates semantic description by extending UDDI architecture. This framework provides both service publishing and service discovery. We use an ontology based approach to capture real world knowledge for semantic service annotation. The prototype we have implemented gives a whole understanding of the discovery model. The relationship between real world concepts is indeed complex. In the future work, we will use OWL as the ontology representation language to describe the semantic relationship between concepts. We will concentrate on using planning techniques to develop automatic service discovery.

7 References

[1] Martin, J.: Web Services: The Next Big Thing. XML Journal 2, 2001. http://www.sys-con.com/xml/archivesbad.cfm

[2] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana: Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001

[3] UDDI white papers. http://www.uddi.org/whitepapers.html

[4] W3C. SOAP Version 1.2 Part 0: Primer. http://www.w3.org/TR/soap12-part0/, June 2003

[5] DAML-S Coalition: DAML-S: Web Service Description for the Semantic Web. Proceeding of The First International Semantic Web Conference, ISWC01, 2002

[6] The OWL Services Coalition: OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.0/

[7] D.L. McGuinness and F.van Harmelen: OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/, August 2003. World Wide Web Consortium (W3C) Candidate Recommendation

[8] Joachim Peer: Proceeding of The First International Semantic Web Conference, ISWC01, 2002

[9] Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K: Importing the Semantic Web in UDDI. Proceedings of Web Services, E-business and Semantic Web Workshop (WES), 2002

[10] Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K: Semantic Matching of Web Services Capabilities. Proceedings of the 1st International Semantic Web Conference (ISWC), 2002

[11] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, John Miller: Adding Semantics to Web Services Standards. Proceedings of the International Conference on Web Services, ICWS '03, 2003

[12] D. Fensel. Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Heidelberg, Germany, 2001

[13] M. Uschold and M. Grüninger: Ontologies: Principles, Methods and Applications. In Knowledge Engineering Review, 11(2), 1996

[14] http://www.w3.org/2002/04/17-ws-usecase