Chapter XV
A Survey of Web Service Discovery Systems

Le Duy Ngan
Nanyang Technological University, Singapore

Angela Goh
Nanyang Technological University, Singapore

Cao Hoang Tru
Ho Chi Minh City University of Technology, Viet Nam

ABSTRACT

Web services form the core of e-business and hence, have experienced a rapid development in the past few years. This has led to a demand for a discovery mechanism for web services. Discovery is the most important task in the web service model because web services are useless if they cannot be discovered. A large number of web service discovery systems have been developed. Universal Description, Discovery and Integration (UDDI) is a typical mechanism that stores indexes to web services but it does not support semantics. Semantic web service discovery systems that have been developed include systems that support matching web services using the same ontology, systems that support matching web services using different ontologies, and systems that support limitations of UDDI. This paper presents a survey of web service discovery systems, focusing on systems that support semantics. The paper also elaborates on open issues relating to such discovery systems.
INTRODUCTION

Web service technology enables e-business and e-commerce to become a reality. It has become a competitive tool of companies by reducing cost through fast, effective, and reliable services to customers, suppliers, and partners over the internet. It enables more efficient business operations via the web and enhances business opportunities to companies. These are achieved through its support of discovery, composition, invocation, monitoring and so on. A web service is a software component representing a specific business function that can be described, published, and invoked over the network (typically Internet) using open-standards.

A Web service based on Web Service Description Language (WSDL) (Walsh, 2002) can be termed “non-semantic web services”. However, using WSDL to describe the services only allows them to be accessed by keyword. This limitation prevents fully automatic discovery, composition, invocation, and monitoring. The reason for this shortcoming is the lack of semantic understanding. To overcome this problem, web services require a method to incorporate semantics. Just as the Semantic Web is an extension of the current World Wide Web, a semantic web service is an extension of web services. It overcomes web service limitations by using knowledge representation technology from the semantic web. Specifically, it uses ontologies to describe its service instead of using WSDL. Such ontologies can be understood by machines and can be reasoned upon. This allows a fully automatic discovery, composition, invocation, and monitoring in web services.

In a web service model, a service provider offers web services which provide functions or business operations which can be deployed over the Internet, in the hope that they will be invoked by partners or customers; a web service requester describes requirements in order to locate service providers. Publishing, binding, and discovering web services are three major tasks in the model.

Discovery is the process of finding web services provider locations which satisfy specific requirements. Web services are useless if they cannot be discovered. So, discovery is the most important task in the web service model.

The greatest difficulty in a web service discovery mechanism is heterogeneity between services (Garofalakis et al., 2004). Heterogeneities include different platforms, different data formats as well as heterogeneities of ontologies. Regarding ontology heterogeneities, semantic web services may use different ontologies or different ontologies description language such as OWL, DAML, RDF etc to describe the services. There is also heterogeneity between semantic web services and non-semantic web services. Therefore, when developing a discovery system, these heterogeneities should be borne in mind.

A survey of web service discovery system is needed to explore existing techniques and to highlight the advantages and disadvantages of each system. (Garofalakis et al., 2004) presented a survey on these systems but their work mainly focused on aspects and approaches of web service architecture and has not paid adequate attention to the usage of semantics. In semantic web services, the usage of semantics is the most important factor. This paper presents a survey of web service discovery systems which focuses mainly on the use of semantics. In this paper, we use the term “non-semantic web services” to refer to web services without semantics, whereas the term “web services” is used in a generic manner to cover both semantic web services and non-semantic web services.

The rest of the paper is as follows. Section 2 introduces the background of web service description languages and the web services model. Section 3 introduces a taxonomy of web services discovery systems. In this section, advantages and disadvantages of each system are highlighted. Section 4 presents the issues related to web services discovery, followed by the conclusion in section 5.
WEBSERVICES DISCOVERY

Web service discovery is widely used in workflow, e-learning, and e-business systems such as e-supply chain, e-manufacturing, etc. Non-semantic web services can be discovered using UDDI (Walsh, 2002). UDDI is an industry specification for describing, publishing, and finding web services. It allows developers to describe and classify their services, and the technical details about the interfaces of the web services which are exposed. UDDI also enables developers to consistently discover services, or interfaces of a particular type, classification, or function. It also defines a set of Application Programming Interfaces (APIs) that developers can use in order to interact with UDDI data directly. The UDDI scheme uses White, Yellow, and Green pages as data categories. White pages store basic contact information about an advertised company. Yellow pages store industry classification. Green pages provide technical information on the behavior and supported functions of a business service hosted by a business.

Semantic web service discovery systems have been developed to discover semantic web services. To be discovered, web services must be described by web service description languages. This section introduces web service description languages, followed by the web service model and several categories of discovery systems.

Web Service Description Languages

WSDL is used to describe non-semantic web services. Ontologies and features such as service name, service description in text, etc. are used to describe semantic web service. The following is an introduction to WSDL and semantic web description languages.

WSDL—A Non Semantic Web Description Language

Web Service Description Language (WSDL) (Walsh, 2002) is a standard to describe how to access a non-web service and what operations (methods) it performs. It is also used to locate web services when a service requester and provider use WSDL to describe the service. A WSDL document defines a web service using four major elements, namely, port, message, types, and binding. An elaboration of these elements is as follows:

1. WSDL Ports: The <portType> is the most important element in WSDL. It defines a web service with operations that can be performed and messages that are involved. The <portType> element is similar to a function library (or a module, or a class) in a traditional programming language.
2. WSDL Messages: The <message> element defines an abstract, typed definition of the data being communicated. Each message can consist of one or more logical parts like parameters of a function call in a traditional programming language.
3. WSDL Types: The <types> element defines a container for data type definitions that are relevant for the exchanged messages. For maximum interoperability and platform neutrality, WSDL uses XML Schema syntax to define data types.
4. WSDL Bindings: The <binding> element defines a concrete protocol and data format specification for a particular port type.

Semantic Web Service Description Languages

Semantic web services use ontologies to describe their services. OWL-S, DAML-S, and RDFS are examples of semantic web services description languages which use OWL, DAML, and RDF ontology respectively to describe their services.
DAML-S and OWL-S which are widely used web service description languages include Service class and three sub-classes, namely, serviceProfile, serviceModel, and serviceGrounding (figure 1).

Matching uses only ServiceProfile because it contains the description of the service to be discovered. The ServiceProfile represents the service through class Profile. A class Profile has three basic types of information, namely, contact information, functional description of the service, and additional properties.

The contact information is meant for human users. It includes serviceName which is the name of the service, and contactInformation which shows the address, telephone number, etc. of the owner of the service. The textDescription provides a brief description of the service. It summarizes what the service offers, or describes what service is requested. In discovery, textDescription of the web service provider and requester is checked to determine if two services have a common description. If such a commonality exists, it implies that the provider is able to meet the requester’s need.

The functional description of the service is the most important declaration in the Profile that is used for matching. It specifies the parameters comprising the inputs, outputs, preconditions, and effects of the service. The inputs are parameters required by the service and the outputs are the parameters which are generated by the service. The precondition indicates the condition necessary before execution of the service and the effect is the condition after execution of the service. For example, to invoke a web service to buy a computer and pay via credit card, the input is price, the output is the computer configuration, the precondition is that the credit card must be valid, and the effect is that the credit card must be charged. Additional properties which describe features of the service are serviceParameter, qualityRating, serviceCategory, etc. The Profile also declares an operation of the web service.

Web Service Model

The web service model in figure 2 shows the interaction between a service requester, service providers, and a service discovery system.

- The service providers offer web services which provide functions or business operations. They are created by companies or organizations. In order to be invoked, the web services must be described. This will
facilitate discovery and composition. WSDL or service profile of semantic web service is used to carry out this function.

- The web service requester describes requirements in order to locate service providers. Service requesters usually contain a description of the web service, though it is not a web service which can run on the internet. The requirements are usually described by WSDL, service template or service profile.
- The web service discovery or service registry is a broker that provides registry and search functions. The service providers advertise their service information in the discovery system. This information will be stored in the registry and will be searched when there is a request from service requester. UDDI is used as a registry standard for non-semantic web service. Semantic web service discovery systems are developed for semantic web services.

The above three components interact with each other via publishing, discovery, and binding operations. These operations are elaborated upon as follows:

1. **Publish**: the web service providers publish their service information through the discovery system for requesters to discover. Through the publishing operation, the web service provider stores the service description in the discovery system.
2. **Discovery**: the web service requesters retrieve service providers from the service registry. Based on service descriptions, which describes the requirements of the web service requesters, the discovery system will output a list of web service providers which satisfy the requirements.
3. **Bind**: After discovering, the discovery system provides a number of web service providers. The web service requester invokes these web service providers. The binding occurs at runtime. The web service requesters and web service providers will communicate via SOAP protocol (W3C; Walsh, 2002) which is an XML based protocol for web service exchange information.
Category of Web Service Discovery Systems

There are many ways to categorize web services discovery systems. Depending on various view points, we can category them as follows: distributed systems, supporting QoS systems, or applying fuzzy logic to the systems. Details of each category are given in the following subsections.

Distributed Discovery Systems

While web services have provided distributed operation execution, most discovery systems are still based on centralized registries. However, as the number of web services grows rapidly, a single point failure and performance bottleneck result in centralized registries becoming impractical. To overcome the disadvantages of the centralized systems, a large number of decentralized solutions based on P2P technologies have been proposed. Most of the systems build on P2P network use ontologies to publish and discover the web services descriptions. Some typical systems are found in (Gagnès et al., 2006; Y. Li et al., 2006; Wu et al., 2005; Yu et al., 2004a, 2004b). These systems present approaches for distributed web services organization by combining the capabilities of semantic web services with the dynamics and real-time search capabilities of peer to peer (P2P) networks. They use category ontology to organize services based on semantic classification of domains.

QoS Discovery Systems

Web service discovery can be divided into two phases. The first phase is matching functional aspects of the web service such as input, output, precondition, effect, operation ... etc. The output of the first phase may be many advertised web service that satisfy the requirement. To reduce the number of advertised web service providers and to find the most appropriate service, the second phase uses QoS. QoS refers to Quality of Service which considers availability, accessibility, integrity, performance, reliability, regulatory, and security. Several discovery systems supporting QoS have been developed such as (Frolund & Koistinen, 1998; Hondo & Kaler, 2002; Ludwig, 2003; Sahai et al., 2002; Zhou et al., 2005a, 2005b; Zinky et al., 1997).

Applying Fuzzy Logic to Discovery

Since the model-theoretic semantics of the languages used in the Semantic Web are crisp, the need to extend them to represent fuzzy data arises, (Chao et al., 2005; Huang et al., 2005; Recuerda & Robertson, 2005) (Mazzieri & Dragoni, 2005; Stoilos et al., 2005). As ontology languages are based on Description Logics which lack the ability to encode and reason with imprecise knowledge, (Stoilos et al., 2005) extended the DL language with fuzzy set theory and provided sound and complete reasoning algorithms for the extended language.

In semantic web service, the method of using semantics is the most important factor. Therefore, our survey of discovery systems focuses on the use of semantics in web service discovery systems.

A TAXONOMY OF WEB SERVICES DISCOVERY SYSTEMS

This section contains a survey of web service discovery systems which focuses on methods using semantics. A taxonomy of web service discovery systems is introduced. The survey also compares different systems to highlight the advantages and disadvantages of each system.

Introduction to the Taxonomy

Figure 3 presents a taxonomy of web service discovery systems, which can be divided into systems for matching semantic web services and systems
for matching non-semantic web services. The semantic web service discovery systems include systems that support matching web services using different ontologies and systems that support matching web services using the same ontology. There are three main approaches for matching semantic web services using the same ontology, namely, dividing matching process into several stages, matching two profiles directly and supporting UDDI.

The system described by (Liang et al.) is an example of non-semantic web service discovery system. The authors introduced a system that can provide semi-automatic discovery and composition by indexing service advertisements. Currently, most systems use UDDI for discovering such services which are usually described in WSDL. Furthermore, these services use WSDL to register in UDDI. UDDI finds the service providers based on WSDL. The shortcoming of this system is a lack of semantic understanding. This shortcoming has been explained in earlier sections.

Semantic web services overcome the mismatch of web services. This is done by adding more information to the web service descriptions.

**Discovery of Semantic Web Services Using the Same Ontology**

Almost all web service discovery systems that support matching web services assume the same ontology is used. Some systems divide the matching into several stages; others match two service profiles directly, while some support UDDI.

**Direct Matching**

This category of semantic web services discovery systems matches two service description “profiles” directly instead of dividing the matching into several stages. As mentioned in section 2, the service profile consists of information about the services for matching purpose. In these systems, users are not permitted to manage the degree of similarity of matching. Direct matching
systems are accurate but are time-consuming. Furthermore, they do not allow users to manage the trade-off between performance and quality of matching which is the most important characteristic of internet application.

InfoSleuth (Bayardo et al., 1997; Helal, 1997) is one of the earliest matching agent system. The matching agent enables the querying agent to locate all available agents that provide appropriate services. It supports syntactic and semantic matchmaking. In InfoSleuth, the service capability information is written in LDL++ (Hsu & Zaniolo, 1993; Shaw et al., 1993), a logic deduction language. Agents use a set of LDL++ deductive rules to support inferences about whether an expression of requirements matches a set of advertised capabilities. The system provides syntactic and semantic matching. However, LDL++ is outdated and limited in its expressivity. LDL++ is not a standard supported by W3C. At the point of its development, OWL-S and DAML-S which are currently emergent web services description language were unavailable. Therefore, InfoSleuth does not support matching web services using OWL-S and DAML-S.

Similarly, (Trastour et al., 2001) did not use DAML-S or OWL-S for describing web services because their work was carried out before the existence of these languages. Therefore, they developed their own specification for web services description by converting services to RDF graph. Services are nodes of the graph. Two nodes are considered matched if one node is a subtype of the other. Trastour et al extended this work by using DAML+OIL as the web description language (Castillo et al., 2001). The matching is based on the subsumption relationship to find matches between concepts. DL (Description Logic) reasoner is used to determine the similarity between concepts. The system has the drawback that it only considers the relationship between direct sub-concepts and super-concepts. It does not consider the relationship between concepts that are more distantly related.

(Benatallah et al., 2005) present an approach for web service discovery based on Description Logic. They formalized service discovery based on rewriting concepts using terminologies. This is called the best covering problem. They presented a formalization of the best covering problem in the framework of DL-based ontologies and proposed a hypergraph-based algorithm to effectively compute best covers of a given request. In this system, matchmaking is based on DAML-S. Similarly, Li and Horrocks (L. Li & Horrocks, 2003) developed a framework for matching web service based on DAML-S. Their Matchmaker uses Racer (Haarslev & Möller, 2001) as DL (Description Logic) reasoner to determine the semantic match between services. It tries to match two profiles of the requested service and advertised service, instead of dividing the matching into several stages. As mentioned in section 2.1, the service profile is part of any web service description language and includes information such as input, output, precondition, effect, user information etc. Therefore, without dividing the matching into several stages, it will be difficult for these systems to automate reasoning techniques to compute semantic matching and the matching will be inefficient.

Web Service Modeling Ontology (WSMO) provides a conceptual model for service discovery that exploits WSMO formal descriptions of goals and web services. WSMO Discovery Engine (Roman et al., 2004) can match WSMO goals, and hence, web services requester description against web service provider’s description. The advantages are that WSMO can be described semantically. However, WSMO is not a standard of W3C and the current version only supports goal matching.

Dividing the Matching Process into Several Stages

The challenge of dynamic matchmaking in the Internet is to optimize the trade-off between
A Survey of Web Service Discovery Systems

performance and quality of matching. Complex matching has to be restricted to allow meaningful semantic matches of requests and advertisements in a reasonable time. Direct matching systems are accurate but time-consuming. They do not allow users to manage the trade-off between performance and quality of matching. To overcome this, some systems divide the matching process into several stages. Systems that divide the matching process into stages include TU-Berlin (Jaeger et al.), LARKS (Sycara et al., 1999a, 1999b; Sycara et al., 2002), and the Matchmaker from the collaboration between Toshiba and Carnegie Mellon University (Kawamura et al., 2003, 2004).

As heterogeneous web services use different description languages to describe the services and do not understand each other across distributed networks, the LARKS (Language for Advertisement and Request for Knowledge Sharing) project (Sycara et al., 1999a, 1999b; Sycara et al., 2002) defined an expressive common language called ACDL (Agent Capability Description Language). The LARKS matching agent uses this language to match web services requesters with web service providers. It includes five different filters, namely, context matching, word frequency profile comparison, similarity matching, signature matching, and constraint matching. Each filter has different functions. LARKS is expressive and capable of supporting inferences. Its knowledge is specified as local ontologies in the concept language ITL. It supports matching syntactical and semantic similarity among web services by using techniques from information retrieval and AI. It also allows users to restrict the result of matching. However, all services in LARKS must be described in ACDL which is not a standard and is quite difficult to write. Furthermore, LARKS does not support other commonly used web service description languages such as DAML-S and OWL-S.

(Paolucci et al., 2002) have developed an algorithm for matching semantic web services using DAML-S. Their work considers only input and output matching. Input and output are concepts based on an ontology. Therefore, the similarity of input, output matching is treated as the similarity of concepts from the same ontology. Paolucci et al. defined four levels of similarity: exact, plug in, subsumes, and fail. This definition is based on concepts that have a direct relationship: sub-concept or super-concept. Matching two concepts should also include the properties of the two concepts. It should consider general and specific concepts when these concepts are not related directly as sub-concept or super-concept. It also should consider syntactic matching which is based on service name and description of the service.

The semantic web service matching system from the collaboration between Toshiba and Carnegie Mellon University (Kawamura et al., 2003, 2004) is based on LARKS and the algorithm from (Paolucci et al., 2002). To overcome the drawback that requires ACDL, the system supports semantic web services which are described in RDFS, DAML-S, and OWL-S. Similar to LARKS, the system provides a set of filters and allows users to configure these filters to achieve the desired tradeoff between performance and matching quality. There are four filters which are independent of each other. These are Namespace Filter, Text Filter, I/O Type Filter, and Constraint Filter. The system can be installed as an add-on to UDDI; therefore, it supports matching of both semantic web services and non-semantic web services. When it receives a request for a service, it performs the matching process. If it receives a non-semantic web service request, it redirects the request to UDDI to perform the matching process.

The Matchmaker from TU-Berlin (Jaeger et al.) is a good example of a matching algorithm which is divided into four stages. The four stages are input matching, output matching, profile matching, and user-defined matching. By dividing the matching algorithm, the Matchmaker not only avoids the drawback in the system by (L. Li & Horrocks, 2003) but also helps users to choose the degree of matching from each stage in order to control the number of results and accuracy. An earlier version
supported matching DAML-S semantic web services while the current version supports matching OWL-S semantic web services. The user-defined matching stage allows users to describe more constraints to restrict the number of results and to increase the accuracy of the results. However, the system is too simplistic. It only presents a demonstration of matching two web services; it is not really a discovery system since it does not store the advertisement into a database. It is also very difficult for users to utilize user-defined matching. With user-defined matching, the user must define their rules and constraints without user-friendly graphical user-interfaces.

Supporting UDDI

Using UDDI for discovery has limitations. UDDI uses XML to describe its data model. XML’s lack of explicit semantics proves to be an additional barrier to the UDDI’s discovery mechanism. To overcome this problem, some projects add semantics to UDDI, for example, by including semantic web service profile. The project from Carnegie Mellon (Srinivasan et al., 2004) is a typical example. It has added OWL-S profile to UDDI. Kawamura et al. (Kawamura et al., 2003, 2004) added advertising WSDL to UDDI and advertising DAML-S profile to its Matchmaker. If the requested web services use WSDL, the discovery system will access UDDI to look for suitable service providers. Otherwise, the discovery system will revert to the Matchmaker. By combining the Matchmaker and UDDI, the system can support matching both semantic and non-semantic web services, but in a separate manner. In other words, the two types of web services are not considered as alternatives to each other. Matches are to either semantic or non-semantic web services. Another example is from (Sivashanmugam et al., 2003) who have added semantics to WSDL using DAML+OIL ontologies. It also uses UDDI to store these semantics and searches for web services based on these semantics. By adding semantics to WSDL and UDDI, the project in (Sivashanmugam et al., 2003) achieves sufficient expressiveness to automate the discovery process.

The registration and discovery with UDDI is based on a centralized design whereas web service providers developed their own stores called private registries. As web services support distributed operation execution, this has led to a demand to combine UDDI registry with private registries. (Thaden et al., 2003) have developed a distributed peer-to-peer infrastructure. They created a virtual global registry by connecting all private registries in a P2P network. Furthermore, they also used DAML-S to enhance semantic search capabilities.

The advantages and disadvantages of the three methods that use the same ontology are summarised in table 1:

Discovery of Web Services Using Different Ontologies

The above mentioned web service discovery systems only support matching web services requesters and web services providers who use the same ontology. This assumption implies that if different ontologies are used, matching cannot be carried out. This is a major limitation since web services are heterogeneous, autonomous, and developed independently. It is necessary to discover web services that are based on different ontologies.

Cardoso and Sheth in “Semantic e-Workflow Composition” (Cardoso & Sheth, 2003) have addressed this problem. The system supports syntactic matching, quality of service (QoS) matching, and semantic matching. Syntactic matching involves computing the syntactic similarity of web services requesters and web services providers based on their service names and service descriptions. They use “string-matching” as a way to calculate how closely service names and service descriptions of the two semantic web services resemble each other. QoS is used to match web
services requesters and web services providers based on three important QoS factors, namely, time, cost and reliability. Semantic matching is computed relying on concepts and their properties. Property matching is based on domain, name, and range of properties.

However, the system is a part of a work-flow project (Cardoso & Sheth, 2003). Therefore, this discovery system has been developed mainly to support the project. So, it is limited as a discovery system. It supports user defined degree of similarity but it does not allow users to declare more rules or constraints to restrict the matching result. Semantic matching based on only concepts, and their properties is insufficient. It must also include the sub-concept, super-concept and properties of these concepts. The context of concepts also should be considered. Moreover, it is unclear how to determine whether two ontologies are the same or different.

(Oundhankar et al., 2005) presented an extension of the algorithm in (Cardoso & Sheth, 2003) with two new measures: context and coverage similarity. The discovery technique is based on METEOR-S (Patil et al., 2004) which is a web service discovery infrastructure. This infrastructure provides a facility to access registries that are divided based on business domains and grouped into federations. The discovery algorithm used input, output of web services for matching. The matching is divided into syntactic and semantic matching. The syntactic matching is similar to Cardoso’s work which uses n-gram (Angell & Freund, 1983; Salton, 1988; Zamora et al., 1981) technology. But in semantic matching, the algorithm improved Cardoso’s work by considering the context and the coverage information of concepts.

By considering the concept and the coverage information of the matched concepts, the algorithm is able to find good matches and eliminate false matches. Moreover, based on METEOR-S, indexing and retrieving information in the registries is facilitated. However, the system does not allow users to intervene during the matching process. It also does not allow users to define more rules or constraints to restrict the matching result. This will be a significant problem since the number of web services from matching arising is huge. Furthermore, the approach to measure coverage similarity is incomplete since it is only based on similarity and disjointness relationship.

---

### Table 1. Advantages and disadvantages of three approaches to semantic web discovery based on the same ontology

<table>
<thead>
<tr>
<th>Systems</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching Directly</td>
<td>+ Matches all information of the two web services</td>
<td>- Consumes much time to match all information of the two web services.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Does not support user-definition in each part of matching (e.g. input, output matching, etc)</td>
</tr>
</tbody>
</table>
| Divide the matching process into several stages | + Can specify the degree of the similarity  
+ Can define more constraints to enhance the result. | - Some information of web service may not be matched; therefore, the accuracy may be lower than direct matching. |
| Supporting UDDI          | + Can support both semantic matching and non-semantic matching | - Enhancing semantics to UDDI but not to a full extent.                         |

---
Similarly, (Le & Goh, 2005; Le et al., 2005; Le et al., 2006) proposed an algorithm to support matching web services that use both the same ontology as well as different ontologies. The system improves previous work by checking the relationship between ontologies before the matching process. If the two ontologies are related, the concept similarity is computed as they are in the same ontology. The two ontologies are related if they are formed from an original ontology. Otherwise, they are computed as in different ontologies. For different ontologies, the concept similarity is computed based on five similarity dimensions, namely, syntactic, properties, neighborhood, domain, and equivalent similarity. With the five dimensions, the system improves matching and avoids mismatch.

WEB SERVICES DISCOVERY ISSUES

This section introduces three web service discovery issues. They are: matching semantic web services against non-semantic web services, matching semantic web services using different description languages, and matching semantic web services using different ontologies.

Matching Semantic Web Services Against Non-Semantic Web Services

As mentioned in earlier sections, UDDI and some systems have been developed for matching non-semantic web services. They support searching by service names and service categories. They also support matching by inputs and outputs but are not based on semantics. The development of semantic web services has led to discovery mechanisms for semantic web services. Some semantic web services discovery systems have been developed to meet this need. However, these systems only search for semantic web services, and not non-semantic web services. In short, we lack systems which can support matching both semantic web services and non-semantic web services.

For example, a company A developed web services to sell computers online. At that time, semantic web service technology was not available. Hence, WSDL was employed to describe the services. Another company B, which is interested in purchasing computers, started recently. It developed an ontology in OWL and uses OWL-S to describe their request services. Company A is able to provide services to company B. However, with the current web service discovery, company A cannot be found to match the requirement of the company B because they use different web service discovery languages: company A uses non-semantic web services description language while company B uses semantic web services description language.

Matching Semantic Web Services that Use Different Description Languages

Semantic web services have developed very rapidly because of their contribution to e-business and e-commerce applications. There are a large number of semantic web services based on different description languages, including DAML-S, OWL-S, WSMO etc. Existing semantic web services discovery mechanisms support matching based on specific description languages but they cannot perform matching on different description languages.

Suppose a web service uses OWL-S to describe the service. Another service uses DAML-S to describe the service. OWL-S uses OWLJessKB as a reasoner while DAML-S uses DAMLJessKB as a reasoner. The service provider which is described by OWL-S can meet the requirements of a web service requester which is described by DAML-S. However, these two services cannot be matched using current semantic web services discovery.
systems because the two different reasoners do not interact with each other.

**Matching Web Services Using Different Ontologies**

Current discovery systems are adequate when the web service requester and provider use the same ontology and reasoner to determine the relationship between two services. Unfortunately, most of them do not support the situation where a web service requester and provider use different ontologies. Only one group from University of Georgia (Cardoso & Sheth, 2003; Oundhankar et al., 2005) has addressed this problem. However, there are short-comings which were mentioned in section 3.3. These drawbacks should be addressed. In the real world where the web service requester and provider operate independently, each defines their own ontologies to describe their services. However, a web service provider can provide an exact service to the requester even though both services use different ontologies. Therefore, a discovery system that supports web services using distinct ontologies is necessary. This is an emergent research issue that has yet to be addressed.

Consider the following scenario: Some companies in Singapore provide computer solution services. They define their own ontology in the computer domain to describe their services. A company in Vietnam wishes to buy a computer and it defines its own ontology in the same computer domain to describe its services. There are no global ontologies in the domain and the provider and the requester do not know each other. So, though the two ontologies are within the same domain, they are basically different. Current discovery systems are unable to match the web service requester and provider in such a situation, but obviously, the providers can provide an exact fit to the requester’s demands.

**CONCLUSION**

The paper presented a survey of web services discovery systems which focuses on methods of using semantics. A taxonomy of discovery systems was introduced. The advantages and disadvantages of various systems were highlight. There are two categories of the discovery systems that cater to semantic web services and non-semantic web services respectively. UDDI is a typical registry and discovery system for non-semantic web services. For semantic web services, the systems can be categorized as systems that support matching based on the same ontology or on different ontololgies. In systems supporting the same ontology, we can further classify them as systems that divide matching process into several stages, matching directly two web service profiles, or supporting UDDI.

We have introduced three open issues arising from current web service discovery systems, namely, matching semantic web services against non-semantic web services, matching semantic web services using different description languages, and matching semantic web services using distinct ontologies. A new web services discovery mechanism is required to overcome the three shortcomings. When designing a web service discovery system, the main problem of dynamic matchmaking in the Internet, namely the trade-off between performance and quality of matching should be borne in mind. The heterogeneities of web service also should be considered.

**REFERENCES**


A Survey of Web Service Discovery Systems


Jaeger, M. C., Tang, S., & Liebetruth, C., The TUB OWL-S Matcher.


A Survey of Web Service Discovery Systems


