DISCOVERY

Schahram Dustdar and Christian Platzer
Distributed Systems Group
Information Systems Institute
Technical University of Vienna, Austria
{dustdar, christian}@infosys.tuwien.ac.at

Bernd J. Krämer
Department of Mathematics and Computer Science
FernUniversität in Hagen, Germany
http://www.dvt.fernuni-hagen.de/

SYNONYMS

WS-discovery; DISCO

DEFINITION

The term discovery, as far as Web services (WS) are concerned, refers to the process of finding WS that match certain computational needs and quality requirements of service users or their software agents. More technically speaking, WS discovery mechanisms take a specification of certain functional or non-functional criteria characterizing a service and try to locate machine-readable descriptions of Web services that meet the search criteria. The services found may have been previously unknown to the requester.

HISTORICAL BACKGROUND

Since Web services were introduced in the early nineties, service oriented architectures had to deal with the discovery problem and it still persists. Initially, the possibilities to describe Web services properly were limited. XML remote procedure calls (XML-RPCs), which were used early on to connect WS, offered no proper way of describing a Web service's capabilities and therefore forced system designers to find other ways to publish this information. These early services were typically used to achieve a platform independent communication between remote peers, nothing more. This requirement was met with an XML-structured messaging protocol that evolved later to SOAP, a standard protocol of today's Web technology. For the act of discovering services, the actually used protocol made no difference. Services description files were propagated mostly by artificial means, by sending the file per e-mail for instance. In some cases, the developer of the Web service also worked on the client-side implementation.

A proper service description mechanism was only introduced when application developers realized that Web service technology had to be leveraged to a level that obviated the need of service consumer and provider to interact closely with each other prior to using a service. With the definition of the Web service description language (WSDL), it was finally possible to describe the interface of a WS in a standardized manner. The discovery problem, however, still persisted because no means existed to publish a service description at in a widely known index or index, once the implementation on the provider side was completed.

To overcome this barrier, a first draft of the Universal Description, Discovery and Integration, short **UDDI**, standard was released in 1999. UDDI was designed as a vendor-independent specification of an XML-based registry for service descriptions maintained in the form of WSDL documents. The standard was completed in

2002. Its purpose was to enable service providers and consumers to find each other if service request and service offer matched and to provide information about message formats and protocols accepted by a Web service. Apart from defining data models and registry structure, UDDI was also designed to offer simple search capabilities to help service consumers find Web services. Thus UDDI contributed to solve the discovery issue.
As a WSDL description of a Web service interface just lists the operations the service may perform and the messages it accepts and produces, the discovery mechanism of UDDI was constrained to match functionality only. If several candidate services could be found the service consumer was unable to distinguish between them. Therefore people felt the need to be able to express semantic properties or quality aspects of requested services as well. But search mechanisms taking into account semantics and quality-of-service properties require richer knowledge about a registered Web service than WSDL can capture.
To complement the expressiveness of WSDL and facilitate service discovery, DAML-S, an ontology language for Web services, was proposed to associate computer-readable semantic information with service descriptions. Semantic service descriptions are seen as a potential enabler to enhance automated service discovery and matchmaking in various service oriented architectures. For the time being, however, services widely used in practice lack semantic information because no easy to use or even automated method to attach semantic information to service descriptions exists.

SCIENTIFIC FUNDAMENTALS

The service discovery process encompasses several steps. Each step involves specific problems that have to be solved independently. The following list will discuss these steps in ascending order, beginning with the most generic step.

Enabling discovery

At a first glance, the act of discovering a service description matching a set of terms characterizing a service, resembles a search processes for Web pages. Well-known search engines like Google or Live utilize a crawling mechanism to retrieve links to Web documents and create a index that can be searched effectively as users enter search terms. A crawler just analyzes a given Web page for hyperlinks and grinds through the tree structure generated by such hyperlinks to find other Web documents. For Web services, or more precisely Web service descriptions, the case is similar except for one major difference: WSDL files do not contain links to other services. Approaches to write crawlers that search web pages for possibly published service descriptions will produce very poor results, in general [4]. UDDI registries are just designed to eliminate the need for crawlers or alike. As a matter of fact though, open UDDI registries for public Web services are increasingly loosing importance. Especially after the two largest UDDI registries from IBM and Microsoft were shut down in 2005, the vision of public services suffered immensely. Suddenly the starting point to find a public Web service was lost, leaving the possibility to query common Web search engines for Web services as the only alternative. There are, of course, some other registries but they usually do not implement the UDDi specification, which suggests that UDDI may not be an optimal solution for public service registries.
In a corporate environment, however, the initial discovery step is not a real issue. Web service descriptions can easily be published on an internal Web page or in a UDDI registry and are, therefore, easily accessible from within the institution.

Searching in service registries

Assuming that a comprehensive collection of service descriptions has already be established, the question is how to retrieve the closest match to a user query in an efficient manner.
As the title suggests, searching is usually performed by humans and not by software automatically. The challenge is how to create an index of services such that the addition and retrieval of service descriptions can be achieved accurately and fast. Common information retrieval methods are often used for this purpose, ranging from the vector space model for indexing and searching large repositories to graph theoretical approaches for fast processing of a rich data collection.
More or less every registry-based solution encompasses such a facility. Especially UDDI registries often come with a rudimentary interface to query the database for contained services. Unfortunately, the general structure of UDDI with its tModel component-layout, which serves as a container to store detailed service information,

complicates data retrieval. Furthermore, most UDDI entries do not maintain a complete service description but include links to such descriptions kept elsewhere. But these links could be broken or inaccessible at search time. As a result, UDDI queries are usually processed on the business data related to a service and not the service description itself. This fact alone limits the usability of UDDI-based search mechanisms enormously or more precisely: it leaves most of the index quality in the hand of the users.
This area on the other hand is heavily investigated throughout the research community and several approaches have been presented that aim at improving search capabilities on service collections. Those approaches are mostly designed to handle natural language queries like "USA weather service" and are supposed to provide a user interface for various registry implementations.

Querying repositories
A more detailed form of search in service descriptions is entitled as querying. Unlike direct search, in which a user simply provides a set of search terms, queries are formal expressions using some sort of query language. In the case of relational databases, SQL is typically used as a query language. Through a query expression it is possible to search for a specific service signature in a set of service descriptions. Assume, for example, that a user wants to find a weather service and can provide three bits of information: country, zip_code, and the desired scale for presenting the temperature. Assume further that the user wants to express certain quality requirements. Then, a query expression in a language alike SQL might read as follows:

SELECT description FROM services s WHERE
s.input.COMPOSEDOF(country AND zip_code AND useCelsiusScale)
AND s.response_time < 200ms AND s.downtime < 1%
.

This example also reveals a weakness of service descriptions as their signatures are usually not specified using exact type information such as *city* or *country* but rather basic types like string, integer etc. are used. Hence, it seems more appropriate to search for signatures in terms of basic data types only. But this would likely result in mismatches. Re-considering the query above, a corresponding signature using the basic types [string,integer,boolean] can easily be met by other services. There is no way to distinguish positive matches from unwanted ones without additional information or a richer index. These problems are addressed by introducing semantics and domain knowledge. For the values of response_time and downtime in this example, there already exist approaches that constantly monitor and invoke the services in a registry to create a statistical profile of the quality of such a service (**QoS**).
These approaches require a more sophisticated registry type than UDDI represents and are therefore mostly implemented in research prototypes.

Domain-specific knowledge in service descriptions
Another requirement that has to be met by more powerful discovery mechanisms is domain-specific knowledge about a service. To take on the sample above, a discovery mechanism able to match the terms city, zip_code and temperature with the semantic categories location and weather would select just the intersection of services dealing with location and temperature. Although domain information is semantic information in certain respects, it does not mean that the information has to be provided upon service registration. Certain approaches exist that are able to group service repositories according to their most probable domain. The grouping can, for instance, be achieved by using statistical cluster analysis and discover strongly related service descriptions.
On the other hand, domain-knowledge can also be gained by letting the service provider add this information. In practice however, it proved to be problematic to let users define semantic information for a service. Once, this is due to the fact that a certain amount of domain knowledge is needed by the programmer of the Web service but mostly because the categorization assigned by indexers cannot be validated and could therefore be incorrect. This field, just like the following, is still heavily investigated, e.g. under the heading "faceted search". It addresses a broad spectrum of issues but also bears a high potential for innovation.

Semantic annotations
The next logical step towards enhancing service discovery is a complete semantic description of a Web service. A multitude of approaches exist, in which semantic annotations are used to define the concepts behind operations of

Web services and their input and output messages. Those approaches use the widely known resource description framework (**RDF**) to add custom-designed semantic markup to service descriptions. A good example for such a semantic markup language is called DAML-S. It is a DAML-based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-intepretable form. An example for such a markup is shown in Listing 1.

```
<rdfs:Class rdf:ID="BookstoreBuy">
  <rdfs:subClassOf rdf:resource=
        "http://www.www.daml.org/services/daml-s/2001/05/Process#Process"/>
</rdfs:Class>

<rdf:Property rdf:ID="bookName">
  <rdfs:subPropertyOf rdf:resource=
        "http://www.www.daml.org/services/daml-s/2001/05/Process#input"/>
   <rdfs:domain rdf:resource="#BookstoreBuy"/>
   <rdfs:range rdf:resource="#rdfs:Literal"/>
</rdf:Property>
```

Listing 1: DAML-S example

This very short listing basically shows a Web service called `BookstoreBuy` and defines it as a process and therefore as a subclass of the class Process in the corresponding ontology. The second part shows an input to `BookstoreBuy` which is it is a subproperty of the property input of Process, from the process model. With this example, some of the limitations for semantic annotations become more obvious. First of all, the creator of the Web service is required to have additional knowledge about semantic annotations and ontologies. Furthermore, the appended information causes an additional amount of work. Secondly, the ontology used defines sharp boundaries for the level of detail that can be reached through the usage of such annotations. In addition, the problem of misused or erroneous annotations mentioned in the previous step persists. Put together, semantic Web services are seen as a technology with the potential to facilitate more powerful service discovery and machine-readable descriptions. Practical experience, however, showed that an exploitation of semantic information is difficult and still leaves room for further improvements.

Quality-of-service properties
The consideration of quality-of-service (QoS) properties in discovery attempts requires the definition of scales of measurements and metrics to qualify the properties per domain. The scales can be of different kinds including nominal, ordinal, interval or ratio. They are used to assign appropriate QoS property values to a service. Here, a service provider has the choice to associate precise values or just value ranges with service property descriptions. The metrics are needed to rank services that match the functional and semantic requirement of a search according their degree of fulfillment of required QoS properties.
These issues and related modifications to service discovery schemes are still subject to a number of research projects. [6], for example, proposes a services discovery approach in which functional, semantic and QoS requirements are taken into account by iteratively applying related filters.

KEY APPLICATIONS

Some of the concepts presented above, especially the first, more general layers are already used in real world implementations. Service registries, especially those not strictly conforming to the UDDI specification are the main area of application for innovative discovery mechanisms. Search and matchmaking on the other hand is particularly required by IDEs. Especially service composition environments enormously benefit from fast and exact discovery mechanisms. Finding substitutes and alternatives for services in compositions is an important topic in service-oriented architectures.

FUTURE DIRECTIONS

Future directions show considerable tendencies towards the semantic web to enhance service discovery for Web services. This fact however, creates a diversion among researchers on this particular area. Some argue that semantic descriptions are too complicated to be of any practical use, while others argue that they are the only way to leverage Web service discovery and search to a point where they can be processed automatically. Both views are valid and which direction proves to be the most promising will be decided by the work that is yet to come.

Some recent works have proposed a recommendation service and suggest to apply it to collaborative web service discovery. Experiments with such solutions are underway in research labs.

URL TO CODE

A Vector-space based search engine for Web services including a clustering algorithm
`http://copenhagen.vitalab.tuwien.ac.at/VSMWeb/`

A Web service registry with invocation capabilities
`http://www.xmethods.net`

Another Web service search engine
`http://www.esynaps.com/search/default.aspx`

WSBen, a Web service discovery and composition benchmark developed at Penn State
`http://pike.psu.edu/sw/wsben/`

CROSS REFERENCE

Service-oriented architecture (SOA), Web Services, Business process management, Publish/subscribe

RECOMMENDED READING

[1] Christoph Bussler, Dieter Fensel, and Alexander Maedche. A conceptual architecture for semantic web enabled web services. SIGMOD Record, 2002, 2002.

[2] Boualem Benatallah, Mohand-Said Hacid, Alain Leger, Christophe Rey, and Farouk Toumani. On automating web services discovery. The International Journal on Very Large Data Bases, 14(1):84-96, 3 2005

[3] Kokash, N., Birukou, A., D'Andrea, V.: "Web Service Discovery Based on Past User Experience", International Conference on Business Information Systems (BIS), 2007, LNCS 4439, pp. 95-107.

[4] Christian Platzer and Schahram Dustdar. A Vector Space Search Engine for Web Services. In Proceedings of the 3rd European IEEE Conference on Web Services (ECOWS'05), 2005.

[5] Florian Rosenberg, Christian Platzer, and Schahram Dustdar. Bootstrapping performance and dependability attributes of web services: Proceedings of the IEEE International Conference on Web Services (ICWS'06) pages 205-212. IEEE Computer Society, 2006.

[6] Xia Wang, Tomas Vitvar, Mick Kerrigan, Ioan Toma: Synthetical Evaluation of Multiple Qualities for Service Selection, In: 4th International Conference on Service Oriented Computing, pages 1-12, Springer 2006.