OSIAN: Ontology based Service Index Annotator

Viji Gopal, N.S. Gowri Ganesh

Abstract—**The success of web service is to interact with the applications of different domains. To achieve this interoperability, dynamic discovery of web services is essential. The repository of the web service needs an index of the currently available web services on the internet, which can be done by explicit publishing by the developers or by addition of the web services using crawlers. There is a chance that web services are indexed, but in real not available permanently or temporarily due to some reasons. We propose the use of OSIAN (Ontology based Service Index Annotator) to check the web services' availability which can be currently used by the service consumers. This will be of great use for the service consumers as the time taken to search for the web services among a large set of provisioned web services listed out in the repository will be reduced.**

Index Terms—availability, ontology, repository, web services

I. Introduction

The services that are available from the internet using some standard protocols like HTTP or SMTP to access are called web services. Mainly they are of two types - synchronous or asynchronous. Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support. Web services enables a distributed environment in which any number of applications, or application components, can interoperate seamlessly among and between organizations in a platform-neutral, language-neutral fashion. Thus several heterogeneous services can co-exist in a distributed computing environment. Web services support the transparent exchange of documents to facilitate business integration. SOAP, UDDI and WSDL are the three core elements of a web service.

One web service in the internet can make use of another web service in the internet in which the former is called a client and the latter is known as a service. Client needs to locate the other application or a piece of business logic located somewhere on the network. The client queries a UDDI registry for the service either by name, category, identifier, or specification supported. Once located, the client obtains information about the location of a WSDL document from the UDDI registry.

Repositories are a basic part of Web Services. They make it possible to find Web Services. Once a Web Service is found in a repository, the repository also describes how to use the Web Service. The owners of Web Services publish them to the UDDI registry. Once published, the UDDI registry maintains pointers to the Web Service description and to the service. The UDDI allows clients to search this registry, find the intended service and retrieve its details. These details include the service invocation point as well as other information to help identify the service and its functionality. Also UDDI enables companies to advertise the business products and services they provide, as well as how they conduct business transactions on the Web. This use of UDDI has the potential to enable the growth of business-to-business (B2B) electronic commerce.

The rest of the paper is organized as follows. Section II gives the background of the paper. Section III presents the traditional architecture for Web service indexation and discovery. Ontology based Service Index Annotator is discussed in Section IV. Conclusions and future research directions are briefly discussed in Section V.

II. Background

In a normal keyword search the search results are based on match between keywords present in the description of the published services and the search string. Pure keyword based search fails to retrieve services which are described using synonyms of the search string. Moreover, singular/plural word forms used in the service description also affect the search result. The result will contain all the services that contain the word in their interfaces. This

Manuscript received January 7, 08.

Viji Gopal is a Master of Engineering Student in R.M.K.Engineering College, Kavaraipettai, Chennai (e-mail: vijigopalakrishnan@gmail.com).

N.S. Gowri Ganesh is with the Center for development of Advanced Computing, Chennai (phone: 91-44-2461 0880; fax: 91-44-2461-0898; e-mail: nsgganesh@cdac.in).

becomes a stumbling stone in bringing out meaningful results. So using concepts instead of words for matching seems to be a better idea.

A requestor will be looking for some web services that will match his criteria. An efficient search engine is supposed to come up with very useful and specific results such that the requestor can easily select one or more among the returned web services. The search engine gives the user the references to the selected web services, once the user selects one or more web services he can contact the vendors of those web services and acquire the service or services provided by them for his use. A search engine should have the maximum precision and recall to give the best results to the user. If most of the web service references reaching the user are dead or not properly working at that point of time, obviously the user will lose his trust on the web service repository. Here we try to improve the performance of the search engine by the use of ontology and by checking the availability of the web services before they reach the hands of the user.

In relation to computer science, ontology refers to computer-based resources that represent agreed domain semantics. Ontology consists of relatively generic knowledge that can be reused by different kinds of applications or tasks. Computer ontology is said to be an "agreement about a shared, formal, explicit and partial account of a conceptualisation". Domain rules restrict the semantics of concepts and conceptual relationships in a specific conceptualisation of a particular application domain. These rules must be satisfied by all applications that want to use ontology.

"An ontology is a formal explicit description of concepts in a domain of discourse (classes), properties of each concept describing various features and attributes of the concept (known as slots or roles or properties), and restrictions on slots (known as facets or role restrictions). Ontology together with a set of individual instances of classes constitutes a knowledge base." [2]. Semantic Web service technologies, such as the Ontology Web Language for Services (OWL-S), are developing the means by which services can be given richer semantic specifications. Richer semantics can enable fuller, more flexible automation of service provision and use, and support the construction of more powerful tools and methodologies.

There have been a number of research efforts along this track. Birgit Hofreiter *et al* [2] discussed in their paper the current status of ebXML and identified the open research issues to be solved to enhance the commercial application of ebXML. They presented the current status of the repository and enlisted the issues faced by the registry and repository as well as the other components of the ebXML registry. Colin Atkinson *et al* [1] studied the repository in detail and suggested some ideas for improving the existing standards.

The work in this paper is mainly based on the initial work by Eran Toch *et al* [6]. It is a Web-based search engine that uses semantic methods for precise and efficient retrieval of Web services, based on their WSDL descriptions. The approach relies on ontologies as a mean for increasing the certainty of matching service functionality. By using data integration and conceptual model techniques, WSDL documents are analyzed and transformed into a generic service model. Following that, the service properties are mapped to concepts that belong to ontologies in different domains. More specifically, we propose an architecture that will facilitate the discovery of semantic Web services with availability checking.

III. Traditional System

In a traditional system (figure 1), service providers describe the interface to their web service using WSDL, and publish their services in a UDDI repository by providing appropriate "meta data" such as provider identity (white pages), a categorization of the provider's industry (yellow pages) and technical information necessary to invoke the service (green pages). Developers interested in using web services are then meant to be able to find components suitable for their needs by browsing the registry or using the keyword-based UDDI search facilities.

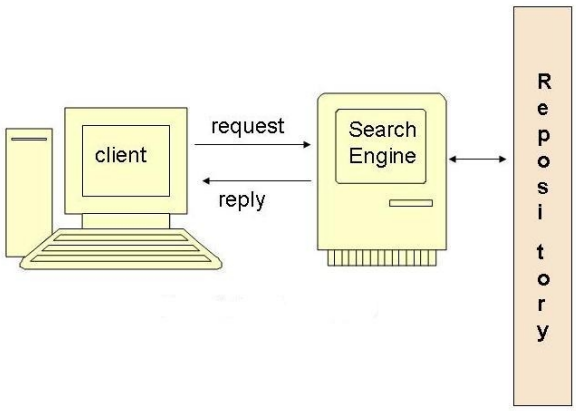

Figure 1: traditional system

To make the search much context specific and meaningful, semantic techniques can be made use of. A web crawler collects semantic web service descriptions from the ontology-oriented UDDI registries, Web sites hosting these services etc and creates and maintains a semantic web service repository. Once a service request is received from a client, the broker in the system can invoke a matching algorithm and a set of web service references are returned to

the user.

IV. Ontology Based Service Index Annotator

A.The architecture

Ontology based Service Index Annotator (OSIAN) is a module that can work in association with a web service search engine. It acts as a mediator between the user and the search engine. Its main purpose is to ease the job of the search engine and to give quick results to the user. OSIAN comprises of two parts:

1. An availability checker
2. RTub (Recent Tub)

The availability checker checks whether a web service is still alive. If yes, it adds the service to the RTub. The web services which are in the RTub as well as in the search engine output will not be checked by the availability checker. RTub is a tub of web services which have been recently checked for availability. RTub is an ontology which has different classes for various domains, for example travel, conferences, pay rules etc. Based on the user input, the services from the corresponding domain is forwarded to the user. Figure 2 depicts the architecture of Ontology based Service Index Annotator.

Once a service is added to the RTub, it will automatically removed from the tub after a specified amount of time that is considered to be a reasonable time in which chances are very less for a web service to shut down. This time interval is called the *lifetime of a service entry* in the RTub. If at all a web service shuts down just after it has been entered in the RTub, it will remain there only for this time interval and it will be quickly removed when its life time expires.

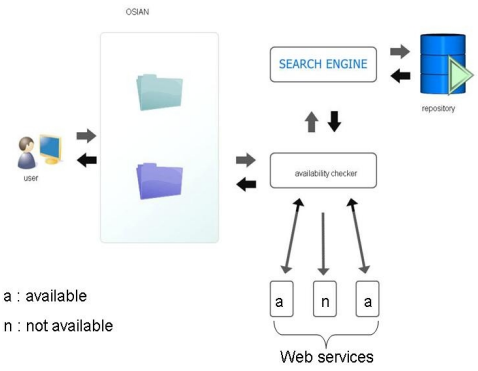

Figure 2: The architecture of OSIAN

When a user gives a query, we must search for specific operations that are similar rather than similar web service names. Even though the web service names are similar, their services may differ. So user comes to know whether a web service is of any use only when he checks it in detail. Two operations are considered to be similar if they take similar input, give similar output and the inputs and outputs have similar relationship between them. This brings out several web services which are of interest to the user. Parameter names in an operation can be grouped into meaningful concepts when a search is invoked. It improves the precision and recall.

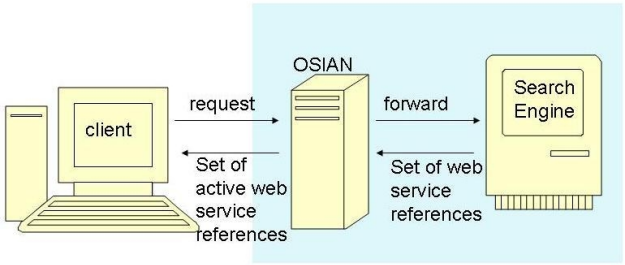

Figure 3: the role of OSIAN in the web service retrieval system

OSIAN will act as a front end of the search engine. It is an intermediate module that connects the user to the search engine. But OSIAN will use the input user interface and output user interface of the search engine. Client posts the search criteria to the interface from which OSIAN absorbs it. Now OSIAN searches its RTub for matches. The ontology matching technique is used here. [6] If matches are available, it is returned to the output interface.

Else the search engine is invoked to search for matching web services in its repository. The search gives an output which is the input of OSIAN. Now OSIAN checks for the availability using its availability checker. The available web services are inserted in the RTub under the proper category. Now they are returned to the output interface so that the user can view them and make a selection from among the set of web services that he/she is presented with. Figure 3 shows how Ontology based Service Index Annotator connects the client to the search engine.

We can test the availability of a potential web service by invoking one of its methods using randomly generated test data. This can be stored in a database and used for later periodic re-evaluation of the web service's availability. The receipt of any valid SOAP response can be interpreted as an indication that the service is at least responding and can be regarded as being "live" [1]. Figure 4 shows the structure of Ontology based Service Index Annotator system.

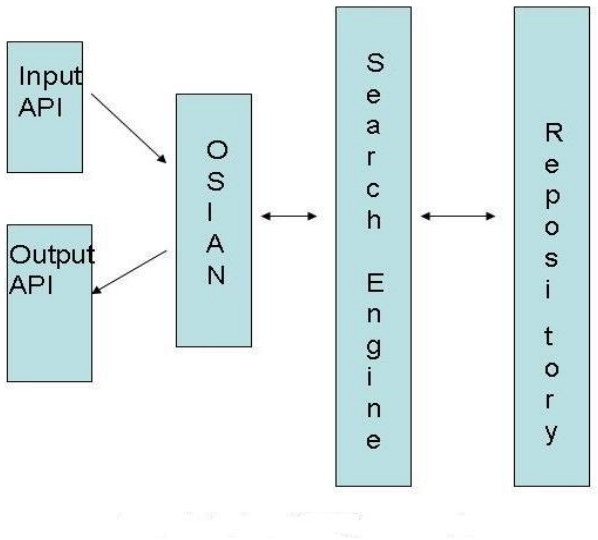

Figure 4: The Structure of OSIAN

B. A simple algorithm

A simple algorithm that describes the availability checking of OSIAN is as follows:

Let *x* be a web service reference returned by the search engine for the repository.

Input: a web service reference from the search engine

Output: the information that the web service is live or not.

Algorithm IV. 1

```
Get x
Check availability
If available
                    Add in RTub
                    Return "available"
Else
    Return "unavailable"
```

The attributes of the web service in RTub are the following:

1) Name[list of parameters]
2) A timestamp

The name is the name or reference to the service. It is the same as in the repository. The timestamp is nothing but the time of availability check on the web service. The system time at the time of availability checking can be used as the timestamp. When the difference between this timestamp and the current system time is equal to the specified lifetime of the web service entry, the web service entry expires and is removed from RTub. This ensures the least number of dead or non-working web service references in the search result.

We can illustrate it with an example. The user gives the input "*input : Car output : Cost"* to know the price of a car to purchase it. A normal search engine searches the repository using some matching algorithm and comes up with a number of web services which have *Car* and *cost* in its description. It may contain web services that calculate the market price of the car, the raw material cost of the car, the human resource cost involved etc. Some of them may be dead too. When the same input is given to Ontology based Service Index Annotator, it searches RTub and checks any web services are there under the *car-cost category* of *ontology* models in the RTub. If it finds such web services, that set is immediately returned to the user. As the web services have been checked for liveness recently, the user is sure to get live web services. If the matching web services are not available, the search is extended to the repository. A matching algorithm finds out matches and returns the result to the availability checker which in turn checks for the availability of each of the web services and returns only those that are *live*. This result is then updated in the RTub and forwarded to the user. Again it is assured that all the services that reach the user are *live*. A web service *W* that was once removed from RTub can be placed in RTub at a later time when a user queries for that kind of web services and *W* is still *alive*.

C. Working of OSIAN: A swimlane diagram

The swimlane diagram of the message exchange in OSIAN is shown in figure 5. There are two cases to be considered: In case 1, we will describe the case where the RTub has the data required by the user. As it already has the data ready with it, it can directly supply the data to the user. The web services in the RTub need not be checked for availability. So the time for availability checking can be saved and the user gets quick reply.

In case 2, we describe the case where the RTub does not have the data required by the user. Now the control is handed over to the search engine and the usual search procedure is carried out. The result is passed to OSIAN and it checks the availability of the web services in the result returned by the search engine. Available web services get qualified to enter the RTub and to be displayed to the user.

OSIAN has the following advantages:

1. For a highly demanded area, OSIAN decreases the number of availability checking by a large amount
2. Almost all web services that reach the user are active. This increases the user's trust on the repository and avoids any wastage of time spent on non-working services.

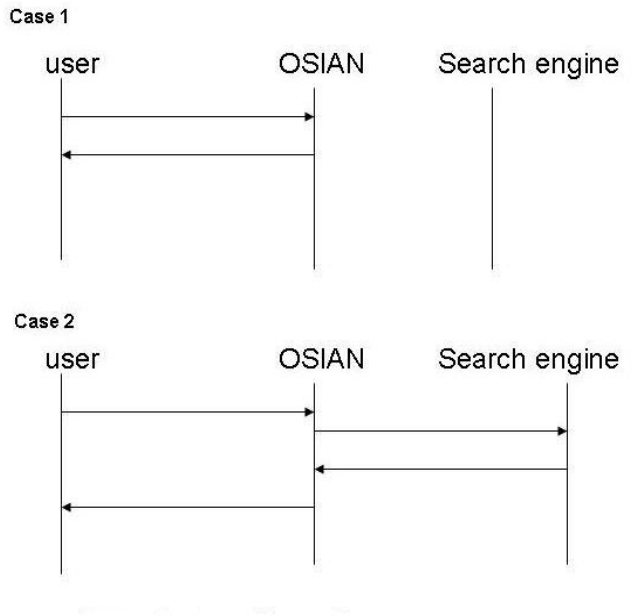

Figure 5: Swimlane diagram

V. Conclusions and Future Research

In this paper, we give an overview of the existing system for web service retrieval. We propose the use of an ontology based annotator to increase the searching speed of a web service repository. It is proved that OSIAN can improve the performance by a great deal by giving a cache effect to the retrieval part of the search engine and by choosing references to more context specific web services. It enables a repository to keep the user satisfaction high by ensuring that all the web services reaching user's hands are alive. There are several directions for future work. At present OSIAN uses the user interfaces of the search engine associated with it. We plan to create a friendly user interface to OSIAN. Also variations of matching algorithms are under trial to give better outputs.

Another issue that comes to light is that UDDI limits the service discovery only to functional requirements. Different web services that satisfy the same requirements can have different Quality of Service (QoS). We can improve the service discovery if we can retrieve services with high QoS. We plan to incorporate this concept as our future work. We can represent QoS specifications using ontology based on XML which enables services to be matched semantically and dynamically. This will allow the user to match the provider's advertised value of a QoS to their own preferences.

References

[1] Colin Atkinson, Philipp Bostan, Oliver Hummel and Dietmar Stoll. *A Practical Approach to Web Service Discovery and Retrieval*. 2007 IEEE International Conference on Web Services (ICWS 2007). July 2007.

[2] Birgit Hofreiter, Christian Huemer, Wolfgang Klas. *ebXML: Status, Research Issues, and Obstacles*. Proceedings of the 12th Int'l Workshop on Research Issues in Data Engineering: Engineering e-Commerce/ e-Business Systems (RIDE'02).

[3] Atkinson, C. and Stoll, D. and Acker, H. and Dadam, P. and Lauer, M. and Reichert, M.U. (2006) *Separating Per-client and Pan-client Views in Service Specification.* In: Proceedings of the 2006 International Workshop on Service-oriented Software Engineering, 27 - 28 May 2006, Shanghai, China. pp. 47-53.

[4] Eyhab Al-Masri and Qusay H. Mahmoud. *Interoperability among Service Registry Standards*. Published by the IEEE Computer Society. IEEE INTERNET COMPUTING. May-June 2007.

[5] Joseph Chiusano (Booz|Allen|Hamilton). *UDDI and ebXML Registry: A Co-Existence Paradigm*. March 2003.

[6] Eran Toch, Iris Reinhartz-Berger, Avigdor Gal, and Dov Dori. *OPOSSUM: Bridging the Gap between Web Services and the Semantic Web.* Proceedings of Next Generation Information Technologies and Systems. July 4-6, 2006. pp. 357-358.

[7] Natalya F. Noy and Deborah L. McGuinness (2001) *"Ontology Development 101: Guide to Creating Your First Ontology"* http://protege.stanford.edu/publications/ontology_development/ ontology101.html

[8] Farquhar, A. (1997). Ontolingua tutorial. http://ksl-web.stanford.edu/people/axf /tutorial.pdf

[9] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein,Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, Katia Sycara. *Bringing Semantics to Web Services: The OWL-S Approach.* www.daml.org/services/owl-s/OWL-S-SWSWPC2004-CameraReady.doc

[10] Wenli Dong. *QoS Driven Service Discovery Method Based on Extended UDDI.* Third International Conference on Natural Computation (ICNC 2007). pp. 317-324

[11] Xin Dong Alon Halevy Jayant Madhavan Ema Nemes Jun Zhang. *Similarity Search for Web Services.* Proceedings of the 30th VLDB Conference, Toronto, Canada, 29 August - 3 September 2004. Pages: 372 – 383

[12] Eran Toch, Avigdor Gal, Iris Reinhartz-Berger, and Dov Dori, *A Semantic Approach to Approximate Service Retrieval.* ACM Transactions on Internet Technology, Vol. 8, No. 1, pp. 2:1-2:30, November 2007.

[13] Berners-Lee, T., Hendler, J., Lassila, O., The Semantic Web, Scientific American, 284(5), 2001, pp. 34-43