AN INTELLIGENT FRAMEWORK FOR IDENTICAL SERVICE EXTRACTION

K. Sendil Kumar*

M. Sakthi Sindhu**

V. Sivaranjani***

T.K. Sri Shilpa****

ABSTRACT

The wide adoption of web services raises the challenging problem of service discovery. In this paper, we propose an agent based framework that discovers similar services based on user's interest. The major challenge in service discovery is to develop a mechanism that finds the service that best suits the user's interest. Due to increase in the number of services available, the search space is so large that to find a service is time consuming. So, we are going to deploy agents, which will act on behalf of the user. Agents possess datastore which are used to perform the discovery process and store the result in its datastore, which reduces the time consumed. Since agents are implemented, there is no need for the client to search for the services each time from the UDDI registry; all the similar services are stored in the datastore.

Keywords: *Service discovery; Web Service Agent; UDDI registry; datastore.*

*Assistant Professor, Sri Manakula Vinayagar Engineering College, Madagadipet.

**Sri Manakula Vinayagar Engineering College, Madagadipet.

***Sri Manakula Vinayagar Engineering College, Madagadipet.

****Sri Manakula Vinayagar Engineering College, Madagadipet

I. INTRODUCTION

A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed. A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services. The technology of Web services is the most likely connection technology of service-oriented architectures. The SOA is the basic architecture used by most RPC-style middleware systems.

A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format. Web Services are modular, self-describing, self-contained applications that are accessible over the Internet. They are based on Open Standards and are Language and Platform Independent.

Web services mitigate the application integration crisis. They help you integrate applications, and they do so at a significantly lower price point than any other integration technology. Web services represent a new form of middleware based on Extensible Markup Language (XML) and the Web. XML and the Web help solve the challenges associated with traditional application-to application integration. Web services simplify the process of making applications talk to each other. Simplification results in lower development cost, faster time to market, easier maintenance, and reduced total cost of ownership. The bottom line is this: Web services allow you to integrate your applications at a fraction of the cost of traditional middleware.

Traditional RPC-style middleware, such as RPC, CORBA, RMI, and DCOM, relies on tightly coupled connections. A tightly coupled connection is very brittle, and it can break if you make any modification to the application. Tightly coupled connections are the source of many a maintenance nightmare. In contrast, Web services support loosely coupled connections. Loose coupling minimizes the impact of changes to your applications. A Web service interface provides a layer of abstraction between the client and server. A change in one doesn’t necessarily force a change in the other. The abstract interface also makes it easier to reuse a service in another application. Loose coupling reduces the cost of maintenance and increases reusability.

Several essential activities need to happen in any service-oriented environment. They are:

- A Web service needs to be created, and its interfaces and invocation methods must be defined.
- A Web service needs to be published to one or more intranet or Internet repositories for potential users to locate.
- A Web service needs to be located to be invoked by potential users.
- A Web service needs to be invoked to be of any benefit.
- A Web service may need to be unpublished when it is no longer available or needed.

The web service uses internet protocols like Universal Description Discovery and Integration (UDDI), Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP) for communication with other web services and agents. SOAP is an XML based protocol which performs data transfer between heterogeneous web services. It enables web site to integrate services from other sites. It is a platform independent and language neutral protocol. WSDL is an XML based language for describing the network services offered by service providers. It describes Web services in a common XML grammar. WSDL specifies how to access the service. UDDI is a standard designed to provide a searchable directory of business and their web services. Service Providers can publish information about their business and the services that they offer in the UDDI. It is a registry of services.

The current Web services architecture encompasses three roles: Web service provider, Web service consumer and Universal Description, Discovery and Integration (UDDI). The Web service provider publishes a description of the service in the UDDI registry. Users (service consumers) will search that directory to get their desired services. In UDDI, information about businesses and services is described in XML.

There are three major operations that surround web services. They are: publishing, which is making a service available, finding, which is locating web services and binding, which is using the web services.

The capability of automatically detecting a service hardware or software from a network is called service discovery. The discovery system either has a centralized repository where services are registered UDDI in case of Web services or they device a mechanism to query all the services in the network. The requirements of service discovery are:

- Descriptions must be attached to different resources (services and workflows) published in different components (service registries, local file stores or databases)
- Publication of descriptions must be supported both for the author of the service and third parties.

- Different classes of user will wish to examine different aspects of the available metadata, both from the service publisher.
- There is a need for control over who make add and alter third party annotations.
- We must support two types of discovery: the first using cross-domain knowledge; the second requiring access to common domain ontology.
- A single, unified interface for all these kinds of discovery should be made available to the user.

Web service discovery is based on matching abstracted goal descriptions with semantic annotations of web services. This discovery process can only happen on an ontological level i.e., it can only rely on conceptual and reusable. Similar service discovery is the process of discovering set of services that are similar in nature to the service in hand is another challenging issue that has be addressed in recent year. Matching is an important in aspect in service discovery. The challenge lies in providing a best matching method that matches the demand and the offer. Similar to marching ranking of service plays a vital is discovering the service that best suits the demand constrain. A ranking model based on quality is a challenging task in web service discovery.

An agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided. A web services agent is a software agent that supports the implementation or access of a web service. Agent is a computational resource. It can also be defined as program acting on behalf of a person or an organization. Software agents are the running program that drives web service, both to implement and to access them.

We propose an agent based similar service discovery framework that considers the QoS parameters requested by the service consumer. The rest of the paper is structured as follows. Section II reviews some related works and our proposed discovery framework is illustrated in section II. Finally, the conclusion and future work are discussed in section III.

II. RELATED WORKS

With an increasing number of Web services providing similar functionalities, Quality of Service (QoS) is becoming an important criterion for selecting of the best available service [1]. QoS is a set of performance and domain-dependent attributes that has a substantial impact on web service requesters' expectations. The discovery process falls in two steps: Web Service matching, which is meeting specific functionality required by a consumer from existing services and Web Service selection which is choosing a service with the best quality

among those matched services. Now, with an increasing number of Web services providing similar functionalities, more emphasis is being placed on how to find the service that best fits the web service consumer's requirements which are functional requirements, and nonfunctional requirements, such as the price and quality of service guaranteed by a service provider.

QoS of a web service is a set of nonfunctional attributes of the entities used in the path from the web service to the client that bear on the web service's ability to satisfy stated or implied needs in an end-to-end fashion [2]. QoS attributes can be distinguished into measurable and unmeasurable ones. A measurable QoS attribute can be measured with the use of one or more QoS metrics. QoS metrics have specific value types, while their values are associated to a specific unit. Moreover, QoS metrics are distinguished between resource and composite ones. Resource metrics are computed directly from the instrumentation of the service's management system, while composite ones are derived from other metrics with the help of mathematical formulas. Unmeasurable quality attributes cannot be measured at all. QoS characterizes any entity used in the path from the user to the web service, including the web service's host and intervening network. Therefore, the QoS characteristics of all these entities constitute the QoS of a web service.

If multiple Web services provide the same functionality, then Quality of Service requirements can be used as a secondary criterion for service selection [3]. QoS is a set of nonfunctional attributes like service response time, throughput, reliability, and availability. Response time is the time a service takes to respond to the client request. Throughput is the rate at which a service can process request attributes. Availability represents the probability that a service is available. Reliability represents the ability of a web service to perform its required functions under stated conditions for a specified time interval.

Service consumers can search the registry, find a service they need, and interact with that service. Web services are primarily used over the Internet, and a UDDI registry factors out the Web service's location constraint. At once, a UDDI registry can offer a collection of web services performing a particular task, letting users pick and choose the one that best suits their particular situation. UDDI has become the de-facto standard for registering services. Its advantages are twofold: first, organizations have a trivial registry for all its services regardless of geographic location, business units and departments and second, they can consolidate system assets, saving on business cost and time.

Integrating Web services and software agents brings about the immediate benefits of connecting application domains hosting one or the other technology: A Web service should

be able to invoke an agent service and vice versa [4] [6]. An agent possesses the ability to comprehend and interact with its environment. Several arguments have been made to support the idea of integration of web service and agent infrastructure.

Web services are conceived as an essential component for promoting interoperability of business processes and software agents as a key enabling technology for such processes to be dynamically discovered, combined and executed. The aim of the semantic Web service is to describe and implement Web services so as to make them more accessible to automated agents. The key idea is to represent the functionality of a Web service explicitly, by using the so-called semantic annotations.

According to a set of QoS criteria, Web services are ranked by using some algorithm to find the best suitable Web services [5]. The functional properties describe what the service can do and the nonfunctional properties depict how the service can do it. The current UDDI registries only support Web services discovery based on the functional aspects of services.

III. PROPOSED FRAMEWORK

We propose a framework for agent-based similar service discovery with QoS for an objective of selecting the best web service that satisfies consumers QoS constraints and preferences. The framework involves four main participating roles. They are Web Service Consumer (client), Web Service Provider, Similar Service Extractor Agent and UDDI. The Web Service Provider is the entity that develops the web service and describes its functionalities in addition to the QoS it provides. Web service consumer can invoke the web service by searching UDDI registry and obtain the access point of a particular web service. The communication between service consumer, UDDI registry and service provider is done using SOAP.

The similar service extractor agent does the following functions:

- It receives requests from the web service consumer, specifying the service functionality along with the QoS constraints.
- Based on the received requirements specification, it searches for the services in its datastore.
- If the requested service is not present in the datastore, then the agent consults the UDDI and discovers functionally similar web services from the UDDI registry.

Due to similar service extraction, if the first service does not match the user demand, then the next service from the ranked list can be checked for matching. This is the major advantage of our proposed system. Agent based approach for similar service discovery hides the system's

complexity from the clients. Agent is concerned with selecting the most suitable web service which satisfying the consumer's QoS constraints and the specific service functionality.

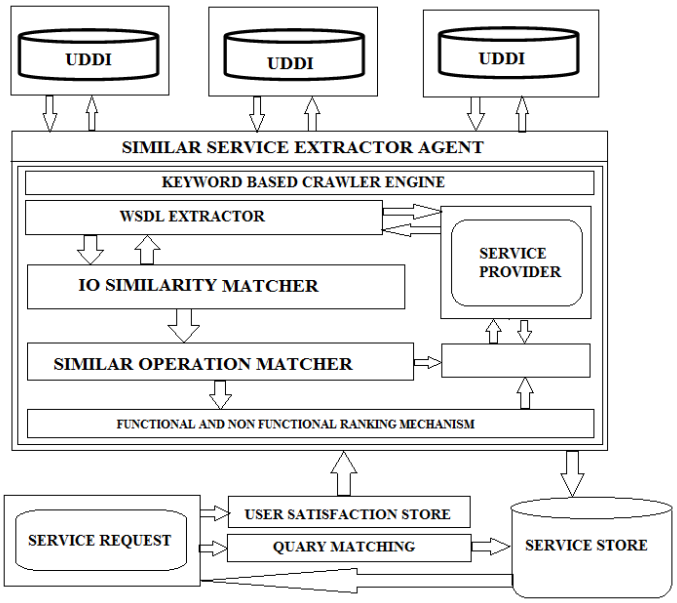

Figure 1. PROPOSED FRAMEWORK

A. QoS parameters for web services

Quality of Service (QoS) is a broad term used to denote the level of predictability and manageability of the services supplied by one or more service providers. It is a set of performance and domain dependent attributes that has a substantial impact on the web service requestor's expectations. Thus it can be used for distinguishing between many functionally equivalent web services that are available now-a-days. QoS support for web services can provide a new business value to service providers by guaranteeing a certain service quality for users. An enhanced QoS for a web service will bring a highly competitive advantage for web service providers.

The QoS parameters will reflect, to a large extent, the eventual preference of service consumers towards a specific web service. Most commonly used QoS parameters are:

1. Availability:

Availability is the quality aspect of whether the Web service is present or ready for immediate use. Availability represents the probability that a service is available. Larger values represent that the service is always ready to use while smaller values indicate unpredictability of whether the service will be available at a particular time. Also associated with availability is time-to-repair (TTR). TTR represents the time it takes to repair a service that has failed. Ideally smaller values of TTR are desirable.

2. Accessibility:

Accessibility is the quality aspect of a service that represents the degree it is capable of serving a Web service request. It may be expressed as a probability measure denoting the success rate or chance of a successful service instantiation at a point in time. There could be situations when a Web service is available but not accessible. High accessibility of Web services can be achieved by building highly scalable systems. Scalability refers to the ability to consistently serve the requests despite variations in the volume of requests.

3. Integrity:

Integrity is the quality aspect of how the Web service maintains the correctness of the interaction in respect to the source. Proper execution of Web service transactions will provide the correctness of interaction.

4. Performance:

Performance is the quality aspect of Web service, which is measured in terms of throughput and latency. Higher throughput and lower latency values represent good performance of a Web service. Throughput represents the number of Web service requests served at a given time period. Latency is the round-trip time between sending a request and receiving the response.

5. Reliability:

Reliability is the quality aspect of a Web service that represents the degree of being capable of maintaining the service and service quality. The number of failures per month or year represents a measure of reliability of a Web service. In another sense, reliability refers to the assured and ordered delivery for messages being sent and received by service requestors and service providers.

6. Regulatory:

Regulatory is the quality aspect of the Web service in conformance with the rules, the law, compliance with standards, and the established service level agreement. Web services use a lot of standards such as SOAP, UDDI, and WSDL. Strict adherence to correct versions of standards by service providers is necessary for proper invocation of Web services by service requestors.

7. Security:

Security is the quality aspect of the Web service of providing confidentiality and non-repudiation by authenticating the parties involved, encrypting messages, and providing access control. Security has added importance because Web service invocation occurs over the

public Internet. The service provider can have different approaches and levels of providing security depending on the service requestor.

8. Scalability:

Web services should be provided with high scalability. Scalability represents the capability of increasing the computing capacity of service provider's computer system and system's ability to process more users' requests, operations or transactions in a given time interval. It is also related to performance. Web services should be scalable in terms of the number operations or transactions supported.

9. Capacity:

Web services should be provided with the required capacity. Capacity is the limit of the number of simultaneous requests which should be provided with guaranteed performance. Web services should support the required number of simultaneous connections.

10. Robustness:

Web services should be provided with high robustness. Robustness here represents the degree to which a web service can function correctly even in the presence of invalid, incomplete or conflicting inputs. Web services should still work even if incomplete parameters are provided to the service request invocation.

11. Exception Handling:

Web services should be provided with the functionality of exception handling. Since it is not possible for the service designer to specify all the possible outcomes and alternatives, exceptions should be handled properly. Exception handling is related to how the service handles these exceptions.

12. Accuracy:

Web services should be provided with high accuracy. Accuracy here is defined as the error rate generated by the web service. The number of errors that the service generates over a time interval should be minimized.

13. Interoperability:

Web services should be interoperable between the different development environments used to implement services so that developers using those services do not have to think about which programming language or operating system the services are hosted on.

B. Non-QoS Parameters

There is also some non-Quality of Service parameters that are to be considered. Some of them are:

1**. Precondition**:

A precondition defines a set of assertions that must be met before a Web service operation can be invoked. They can specify requirements that must be met. Preconditions are specified as child elements of the operation for which the precondition is defined. A precondition is a set of semantic statements that are required to be true before an operation can be successfully invoked.

2. Effect:

An effect defines the result of invoking an operation. It can simply state that the output is returned or it can make statements about what changes in the state are expected to occur upon invocation of the service. Preconditions and effects are primarily used in service discovery, and are not necessarily required to invoke a given service. An effect is a set of semantic statements that must be true after an operation completes execution after being invoked. Different effects can be true depending on whether the operation completed successfully or unsuccessfully.

3. Input:

Input semantics is the meaning of input parameters as defined by some semantic model.

4. Output:

Output semantics is the meaning of output parameters as defined by some semantic model.

C. Working Mechanisms

The main entities involved in the framework are: Service consumer, service provider, UDDI, similar service extractor agent and datastore.

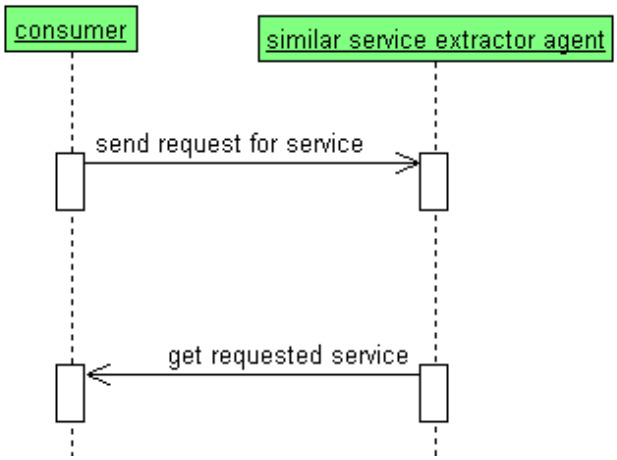

Figure 2. SERVICE REQUEST AND RESPONSE

The initial and final steps are shown in figure 2. The service consumer sends the request for the service needed along with the QoS parameters to the similar service extractor agent. The agent, after receiving the request from the consumer, searches for the service in its datastore at first. If the customer requested service is found in the datastore itself, the agent sends the service to the consumer. If the requested service is not found in the datastore, then the agent searches for the service in the UDDI and then sends the result to the consumer.

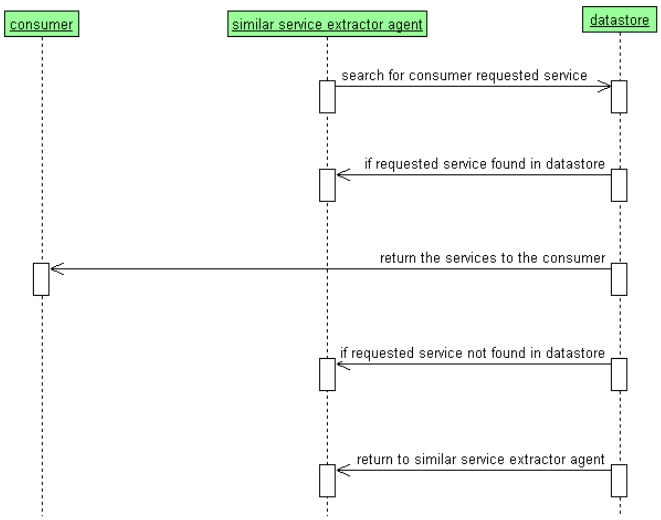

Figure 3. RETRIEVING RESULTS FROM DATASTORE

After receiving the request from the consumer, the similar service extractor agent searches for the consumer requested service in its datastore. If the consumer requested service is found in the datastore, the agent returns the service to the consumer. If the datastore does not contain the consumer requested service, then the agent performs the search process in the UDDI.

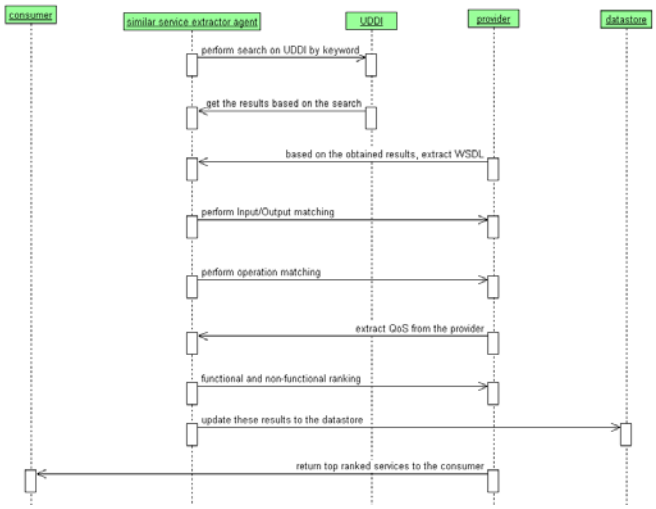

Figure 4. RETRIEVING RESULTS FROM UDDI

If the consumer requested service is not found in the datastore of the agent, the agent performs keyword based search in the UDDI. Based on the content retrieved, the agent extracts the WSDL of the corresponding service from the service provider. It then performs the input, output and operation matching of the services with those of the provider. Then the QoS extraction is done and based on that, the services are ranked. Since these services are not present in the datastore, the results are first updated in the datastore and then sent to the consumer. If the consumer's request is not completely satisfied with the first service retrieved, then the consumer can use the next service from the ranked list till his need is fulfilled. This is the most important feature of our proposed system.

IV. CONCLUSION

In this paper, we have presented a new approach for web service discovery process. Due to the increasing popularity of web services technology and the potential of dynamic service discovery and integration, multiple service providers are now providing similar services. QoS is a decisive factor to distinguish functionally similar web services. Due to similar service extraction, if the first service does not match the user demand, then the next service from the ranked list can be checked for matching. This is the greatest advantage of our similar service discovery framework. An amount of services is needed where we can test the performance of our system. This will enable a more flexible, and trustable architecture.

REFERENCES

[1] T. Rajendran and P. Balasubramanie "An efficient framework for agent based quality driven web service discovery" 2009 IEEE.

[2] Kyriakos Kritikos and Dimitris Plexousakis "Requirements for Qos based web service description and discovery" IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 2, NO. 4, OCTOBER-DECEMBER 2009.

[3] T. Rajendran and Dr.P. Balasubramanie "An optimal agent-based architecture for dynamic web service discovery with Qos" presented at 2010 Second International conference on Computing, Communication and Networking Technologies 2010 IEEE.

[4] Azadeh Ghari Neiat, Mehran Mohsenzadeh, Sajjad Haj Shavalady and Amir Masoud Rahmani "A new approach for semantic web service discovery and propagation based on agents" presented at 2009 Fifth International Conference on Networking and Services 2009 IEEE.

[5] T. Rajendran and Dr.P. Balasubramanie "Flexible and intelligent architecture for Quality based web service discovery with an agent based approach" Proceedings of the International Conference on Communication and Computational Intelligence – 2010.

[6] Azadeh Ghari Neiat, Mehran Mohsenzadeh, Rana Forsati and Amir Masoud Rahmani "An agent based semantic web service discovery framework" presented at International Conference on Computer Modeling and Simulation 2009 IEEE.