Comparative study of mechanisms for discovering the most appropriate web service and proposing an efficient web service discovery mechanism

Chapter 1

Introduction

Web Service Technology (WST)- A Brief Concept

Companies have reorganized businesses using technology advents such as web-enabled business. These businesses have gotten highly promoted due to the ease with which application-to-application communication happens over the internet, the underlying framework being strong support of web service technology.

The fundamental concept is simple – web services allow us to make Remote Procedure Calls (RPCs) against an object over the Internet or a network. Web Services Technology is not the first of its kind to allow us to do this, but it differs from other technologies in its use of platform-neutral standards. For example HTTP and XML allow us to hide the implementation details entirely from the client. The client needs to know the URL of the service, and the data types used for the method calls, but don’t need to know whether the service was built in Java and is running on Linux, or is an ASP.NET web service running on Windows. [97]

A Web service comprises of loosely coupled software components published, located and invoked across the web. A Web service is a means of performing distributed computing. A web service provides either some business functionality or information to other applications through an internet connection. For example,

- A recruiting company is interested in publishing its latest job-openings as a Web Service. Job placement (contracting) companies could be potential subscribers to this Web Service.
An airline reservation system is interested in publishing its latest airfares as a Web Service. Travel agencies could be potential subscribers of this Web Service.

A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be found out by other software systems. These systems may then interact with the Web service in a way specified by its definition, using XML based messages conveyed by Internet protocols.[98]

1.1 Service Oriented Architecture (SOA) and Web Service Discovery

Since businesses have reorganized using the technology, there came a need to have architecture for building business application known as Service Oriented Architecture.

A Service Oriented Architecture (SOA) is architecture for building business applications as a set of loosely coupled black-box components organized to deliver a well-defined level of service by linking together business processes. One of the most important aspects of SOA is that it is a business approach and methodology as much as it is a technological approach and methodology. With SOA, the important business processes such as generating an invoice, calculating an interest rate, converting currency become business services. A business service is a sealed container of software code that describes a specific business process that can be connected to other business processes. One single business service for a given functionality can be used everywhere in the organization and whenever a business policy is changed, it is required to make change at only one place as the same service is used everywhere.

SOA can make it easier and faster to build and deploy IT systems that directly serve the goals of a business. SOA adds predictability and regularity between business rules, policy and software services. Therefore, one of the greatest selling points for SOA is that it can help management know what tasks a particular service is executing and what rules and policies are codified within these services. Being able to track this not only makes software within the company better but also makes corporate governance more predictable and less cumbersome.
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.

Service-Oriented Architecture is a business-driven IT architecture approach that supports integrating your business as linked, repeatable business tasks, or services. SOA helps today’s business innovate by ensuring that IT systems can adapt quickly, easily and economically to support rapidly changing business needs. SOA helps customers increase the flexibility of their business processes, strengthen their underlying IT infrastructure and reuse their existing IT investments by creating connections among disparate applications and information sources.

Service-oriented architecture is not a new concept. The first service-oriented architecture for many people in the past was with the use of DCOM or Object Request Brokers (ORBs) based on the CORBA specification. In these traditional distributed architectures, web services were used to facilitate point-to-point solutions. Hence, web service discovery was not a common concern.

The increasing number of web services available on the web raises a new and challenging problem, the location and discovery of these services. The lack of a proper discovery mechanism is hindering the potential of these technologies.

The growing numbers of web services descriptions are difficult to manage in open environments such as in the Web. The main problem arises due to the fact that hundreds of different web services exists providing thousands of different functionalities. They are built independent of each other at different locations by different people. Discovering a web service that matches the user's requirement is time consuming and tedious.

As the demand for web service consumption is rising, a series of questions arise concerning the methods and procedures to discover the most suitable web service to use. Web service discovery is the process of finding the most appropriate web services needed by a web service requestor.
Comparative study of mechanisms for discovering the most appropriate web service and proposing an efficient web service discovery mechanism

There is a need for dynamic discovery mechanism that will be always up-to-date providing efficient and available web service choices.

Web service discovery mechanisms have a role even more important than web searching, because they facilitate the need for collaboration among various business processes and consumers over widely accepted web standards.

In the beginning of service-oriented computing, finding relevant web services was mainly done by searching through services registries (i.e. UDDI Business Registries or UBRs). Automated web service search engines were not necessary when web services were counted by the hundreds. However, the number of service registries is gradually increasing and web service access points (i.e. WSDLs) are no longer a scarce resource as there are thousands of web services scattered throughout the Web.

Business organizations need to advertise their services in a global environment to potential trading partners and they should also have a way to discover and interact with each other. Service consuming client must be able to find proper web services with less effort than currently required.

As web services have begun to expand across the internet, users need to be able to efficiently access and share web services. Production and interoperability of larger number of web services have lead to the emergence of new standards on how services can be published, discovered or used. Hence, mechanisms are required for efficient selection of appropriate web service instance in terms of quality and performance factors during web service consumption.

1.2 Approaches towards Web Service Discovery

Web service discovery is "the act of locating a machine-processable description of a Web service that may have been previously unknown and that meets certain functional criteria." [97] The goal is to find an appropriate Web service.

Under manual discovery, a requester human uses a discovery service (typically at design time) to locate and select a service description that meets the desired functional and other criteria.
Under *autonomous discovery*, the requester *agent* performs this task, either at design time or run time. The steps in discovering a web service are same in both cases. Only few issues such as interface requirement’s need for standardization and trust have to be considered in this case, as the discovery is automated.

One situation in which autonomous discovery is often needed is when the requester agent has been interacting with a particular provider agent, but for some reason needs to refresh its choice of provider agent, either because the previous provider agent is no longer available, or other reasons.

There are three main approaches [97] for discovering a web service: as a registry approach, as an index approach, or as a peer-to-peer approach. Their differences and purposes are discussed below.

### 1.2.1 The Registry Approach

A registry is an authoritative, centrally controlled repository of services information. Service provider must publish the service information into the registry before that information is available to the service consumers. The registry owner decides who has authority to publish and update the service information into the registry. A company is not able to publish and update the information of services provided by another company. The registry owner decides what information can be published in the registry. Others cannot independently add to that information. UDDI is an example of the registry approach, but it can also be used as an index.

### 1.2.2 The Index Approach

An index is a collection or guide to information published by the service provider and that exists elsewhere. It is not authoritative and information that it references is not centrally controlled. In the case of an index, the service provider describes the service and functional descriptions on the Web, and the index owners collect them without service providers knowledge. Anyone can create their own index. When descriptions are exposed, they can be collected using web spiders and arranged into an index. Multiple organizations may have such indexes. The information contained in an index could be out of date. The information can be verified before use. Different indexes provide different
kinds of information — some richer, some sparser. Google is an example of the index approach.

The key difference between registry and index approach is one of control: Who controls what and how service descriptions get discovered? In the registry model, it is the owner of the registry who controls this. In the index model, since anyone can create an index, market forces determine which indexes become popular.

1.2.3 Peer-to-Peer (P2P) Discovery

Peer-to-Peer (P2P) computing provides an alternative that does not rely on centralized registries and allows Web services to discover each other dynamically. At discovery time, a service requester queries its neighbors in search of a suitable Web service. If any one of them matches the request, then it replies. Otherwise each queries its own neighboring peers and the query propagates through the network until a particular hop count or other termination criterion is reached.

Peer-to-peer architectures do not need a centralized registry, since any node will respond to the queries it receives. P2P architectures do not have a single point of failure, such as a centralized registry. Furthermore, each node may contain its own indexing of the existing Web services. Finally, nodes contact each other directly, so the information they receive is known to be up-to-date. On the contrary, in the registry or index approach there may be significant latency between the time a Web service is updated and the updated description is reflected in the registry or index. The reliability provided by the high connectivity of P2P systems comes with performance costs and lack of guarantees of predicting the path of propagation. Any node in the P2P network has to provide the resources needed to guarantee query propagations and response routing, which in turn means that most of the time the node acts as a relayer of information that may be of no interest to the node itself. This results in inefficiencies and large overhead especially as the nodes become more numerous and connectivity increases. Furthermore, there may be no guarantee that a request will spread across the entire network, therefore there is no guarantee to find the providers of a service.

Further to the above approaches, justification lies in portraying the issues related to these approaches.
1.3 Issues in Web Service Discovery

In today’s global world, every person is looking for cost and time effective services, which can give him/her satisfaction. Thanks to technological development because of which the world has come closer. There are number of software/IT companies which are providing web based services to global customers. Right from travel booking to buying and selling anything, customers do visit web portals very often. Based on the cost-benefit analysis customer makes selection and try to avail the services. On the other hand service provider companies (web portals) in association with IT companies, who develops the services, make efforts to meet the customers’ needs and to satisfy them. However due to technical and non-technical problems, service providers as well as IT companies do find that customers have genuine complaints or grievances which they can or can not solve immediately. Because of this, loosing customers has become a great loss to the service providers. In order to solve this problem, this research has aimed to develop a model of efficient web service discovery mechanism. This will lead to help service engineers of service provider companies and ultimately general customers in making an effective search while logging onto the site for expected service based on certain parameters which will automatically make discovery by giving ranking/priority for cost-effective solution. Web service discovery based on the non-functional aspects (e.g Quality of Service) has become a very important step to help service requestor to locate a desired service. Generally there are two types of service requestors – the human user who will use the services in complex application development or program which automatically sends request and select services for further processing. Many researchers are proposing various models, QoS description languages and frameworks for discovering and selecting an appropriate web service. However, from the literature study some issues which arise and need to be addressed are as -

- The end user’s view has not been focused in their designs and the user support is either missing or lacking in these systems. Without the proper user support, the accuracy of the QoS requests cannot be guaranteed, and without accurate QoS requests, even the best selection model cannot satisfy users’ requirements. Hence there is a need of a user oriented service selection system, which is important mainly for the human-involved service selection.
An assumption that users can formulate requests which precisely reflect their QoS requirements may not be true as a user may not have the knowledge about what the realistic QoS values are. Also if the user requests for a service with randomly picked number for reliability as ‘greater than 95%’, the result could be zero matching services. Decreasing this number by a few percent, we may find some matching services. Because of this kind of difficulty of choosing a right number, it is not reliable for a selection system to assume the accuracy of the QoS requests from users. It is very advantageous if the selection system can assist users to choose the right QoS values.

In many current systems, the user interface design is not given much importance. Different selection models are proposed and then it is assumed that users would have the ability to submit a proper query which will yield appropriate results using the model. The user may need to have the knowledge on ontology, utility functions etc. In reality, many of the users don’t have this kind of knowledge. So we should have a simple and a carefully designed interface to help users formulate the service request.

With current QoS query languages, requestors may not be able to define their requirements in a precise and comprehensive way. For instance, many times the QoS requirement is represented as either a number (e.g. reliability: 95%), or a fuzzy description (e.g. reliability: very good). However, it is also possible that users may have a mixed request – numeric values on some QoS attributes and fuzzy expressions on others. Therefore, the selection model should have the ability to support this kind of request.

Another issue we want to address is the lack of support for defining preference order on QoS attributes, e.g. which quality attributes should be given higher priority if there are more than one services satisfying all the criteria. Hence, it is necessary to define a separate preference order for QoS attributes, which is lacking in many current works.
1.4 Research Hypothesis

A number of web portals are offering web services for customers all over the world. Customers make selection of these web services based on certain parameters of their choice. However, there are certain loopholes in the mechanism of efficient web services discovery.

The proposed research work aims to develop an efficient model for web services discovery mechanism, which will wipe off the weaknesses in the existing web service architecture to satisfy the customers. The proposed model will assist in retrieving web services with desired functionality and provide a flexible tool which will guide the user to choose the right QoS parameter values, formulate precise requirements for these QoS parameters and define QoS parameters preference order or priorities for minimizing the search. The tool will rank the services based on the search criteria specified by the user and thus the most appropriate web service for the user will be found out using the proposed mechanism.

This research work intends to accomplish the following:

Given a list of web services with the similar functionality and different QoS values, this study aims at

1) Proposing a new discovery technique to store and manage QoS information of web services in the registry for ranking and finding the most appropriate web service from the list of published web services in the registry.

2) Designing a Web Service Discovery tool which will assist in –

   a. Publishing web services along with QoS information in UDDI registry.

   b. Requesting web services by specifying functional, QoS and Monitoring requirement along with the priority of QoS.

   c. Extracting monitor score from the service monitor which monitors the services at regular intervals for verifying advertised QoS by the service providers.

   d. Assigning weights to QoS and monitor scores as per the users preference and find the overall score of each service which are functionally matched.
Comparative study of mechanisms for discovering the most appropriate web service and proposing an efficient web service discovery mechanism

e. Ranking the services based on the calculated overall scores and return specified number of top ranked services to the user.

3) Comparing the proposed technique with existing techniques mainly on the relevance of quality of the results which is evaluated based on the degree of similarity between results obtained from the new technique and that of the existing one.

4) Based on the results obtained from above, implementing a web discovery tool with user-centric interface for discovering the most appropriate web service.

1.5 Research Objective

As a large number of web services are proliferating across the internet, end users or client applications need to be able to efficiently access and share web services. Production and interoperability of larger number of web services have lead to the emergence of new standards on how services can be published, discovered or used. Hence, mechanisms are required for efficient selection of appropriate web service instance in terms of quality and performance factors at the time of the web service consumption.

The discovery mechanism should offer a number of capabilities, recognizable at both development and execution time. During development, one may search a web service repository for information about available web services. At execution, client applications may use this repository to discover all instances of a web service that match a given interface in automated way.

The main objective of this research study is to propose a simple mechanism at the level of standards such as WSDL and UDDI which will attempt to select the most efficient web service among possible different alternatives with real-time, optimized and countable factors-parameters. The mechanism aims at minimizing the search of web services by ranking the matched web services based on functional requirements by keyword search and nonfunctional requirement by QoS parameters.

The work aims to examine and analyze the different mechanisms and models for the web service discovery and thereafter attempts to propose the best discovery mechanism for the desired web service.
1.6 Research Methodology

1) The research study approaches the problem defined in section 1.4 in different phases as follows:

**Phase I**

a. Study the existing web service discovery mechanisms to determine their suitability for discovering the most appropriate web service from the available set of services for the desired functionality and find out the suitable ones. A pilot survey study was also conducted whose result signifies necessity of an efficient mechanism which should be able to discover the most appropriate web service as per the consumer’s requirement of functionality as well as quality of service (QoS) and the priority of QoS.

b. Identify the QoS parameters for ranking web services for efficient discovery under the given environmental constraints.

c. Design an algorithm for matching web services with desired functionality based on keyword search and ranking web services based on the QoS parameter values with its preference values specified by the user.

**Phase II**

a. Design an algorithm for calculating monitor ratings and score for each functionally and QoS matched service.

b. Design an algorithm for ranking the web services based on the overall scores i.e. both actual QoS and monitored QoS score.

**Phase III**

Implement the existing and proposed discovery algorithms designed in Phase I and Phase II and investigate the performance of each based on quality of result obtained.
Comparative study of mechanisms for discovering the most appropriate web service and proposing an efficient web service discovery mechanism

2) The research work provides

a. Comparative study of existing web service discovery mechanisms to determine their suitability for discovering the most appropriate web service from the available set of services for the desired functionality and find out the suitable ones
b. Algorithms for matching and ranking of web services for the selection.
c. Result analysis of existing discovery algorithm and proposed discovery algorithm based on QoS parameters.

1.7 Organization of Thesis

In Chapter 2, the review of literature and various mechanisms for web service discovery are presented. It starts with describing the concept of web service discovery and further discusses various web service discovery mechanisms. It presents how the search engines like Google, Yahoo are not useful enough for discovering the services available over the internet, as those searches are generic and it could only locate publicly accessible WSDL documents. Various mechanisms to discover web services have been reviewed and presented. It also presents the impact of centralized mechanisms UDDI and ebXML, on the way of conducting the e-business by making it possible for business organizations to publish information on the internet about their products and web services. Decentralized approaches based on Peer-to-peer mechanisms and federated registry are also discussed.

Chapter 3 discusses about services registries available and the data model of each. Two main services registries are discussed namely, Universal Description, Discovery and Integration (UDDI) registry and Electronic Business XML (ebXML) registry. The chapter discusses the architecture and comparative study of both registries.

Chapter 4 discusses the approach of UDDI based mechanisms for web service publishing and discovery in the registry. It presents Reputation-enhanced web service discovery with QoS and Web service QoS-Certifier based web service discovery. The chapter presents introduction of new role in the architecture of UDDI registry – Reputation Manager and Web Service QoS Certifier.

Chapter 5 discusses the new mechanism proposed, i.e. Smart Web Service Discovery enhanced with QoS Monitor, to discover a web service. A detailed discussion on how this
mechanism can be implemented in current UDDI registry architecture is presented. The chapter provides various algorithms for publishing web services in this registry, matching, rating and ranking these web services according to service consumer’s functional and QoS request.

Chapter 6 provides results and analysis of various experiments conducted. This chapter provides a comparison of results obtained from experiments with different mechanisms. The chapter provides a framework under which different experiments were conducted and lists the parameters chosen for these experiments. The analysis of the results obtained is also provided.

Chapter 7 presents the summary of research work carried out. Certain claims about contribution to the knowledge made by the research are put forward. This chapter draws conclusions and directions for the further research.

Appendix – I lists relevant definitions for understanding of web service architecture and discovery of web services.

Appendix – II presents an ER Diagram for jUDDI database which stores the information about of web services on the server.

Appendix – III provides the formats of questionnaire required for the pilot study to start with the research.

Appendix – IV contains a copy of all the published papers during this research work.