Indian Journal of Science and Technology, *Vol 10(17), DOI: 10.17485/ijst/2017/v10i17/108890, May 2017* *ISSN (Print) : 0974-6846*
ISSN (Online) : 0974-5645

Discovery and Invocation of Web Services using Multi-Dimensional Data Model with WSDL

Sumathi Pawar[1]* and Niranjan N. Chiplunkar[2]

[1]Computer Science and Engineering, Canara Engineering College, Mangalore – 574219, Karnataka, India; pawarsumathi@gmail.com
[2]Computer Science and Engineering, NMAM Institute of Technology, Karkala – 574110, Karnataka, India

Abstract

Objectives: The aim of this research is to discover and invoke the Web services dynamically according to the users' requirement. This system resolves the unknown input parameters' names and invokes the required services. **Methods/ Statistical Analysis**: The methodology used is retrieving the Web Service Description Language (WSDL) of the Web services from the online-search-engine according to the user request, extracting and storing only the required elements in Multidimensional Online Analytical Processing (MOLAP) Cubes, mining MOLAP records to fill the unknown WSDL parameters' name and dynamically invoking the Web services. **Findings:** The WSDL is big in size due to its multi elemental structure and requires huge space and time for storing and processing. Hence MOLAP cubes are used to store only the required elements of the retrieved WSDL. Existing Web Service discovery methods are classified into several categories: Ontology-based approach, UDDI based approach, Semantic-Based approach, QoS-based approach, Information Retrieval Based approach, data mining based approach and link-based approach. The drawbacks of the existing systems motivated the proposed system to use the WSDL-based method and to use an online search engine for the discovery of Web services. The proposed system is designed and implemented by considering the dynamic nature of the Web services. The results of experiments justify that the proposed system satisfies the user requests with reduced consumption of bandwidth, CPU and storage space. The performance of proposed system is high compared to the existing non-MOLAP systems. **Application/ Improvements:** Developing bigger applications with fewer resources and lesser time. Reusability of existing procedures through Remote Procedure Call (RPC). Satisfaction of the user requests at high speed.

Keywords: Data Mining, MOLAP, Service Discovery, Service Invocation, Web Service, WSDL

1. Introduction

Web services are loosely coupled, self-contained software modules that can be accessed programmatically using existing Internet technology.[1] The web services are found and assembled dynamically to serve a particular function, to solve a specific problem or deliver a particular solution to a customer. In a Web services scenario, a web application sends a request message to a service at some URI using the Simple Object Access Protocol (SOAP) over communication protocols like Hyper Text Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP) or File Transfer Protocol (FTP).[2] The service receives the request message, processes it and returns a response message.

The proposed system focuses on the discovery of WSDL of Web services using Bingo search engine and processing the WSDL to find suitable Web services. The Web service discovery using WSDL based method is most popular and supported by both industry and development tools.

During the literature survey it is observed that there are many existing Web service discovery systems which use different approaches like Ontology- based, Universal Directory Description Interface (UDDI) based, Semantic-Based, Quality of Service (QoS) based, Information Retrieval Based, data mining based and linking based.[3-5]

The pitfalls of these systems are discussed in the literature survey in Section 2. These drawbacks and absence

Author for correspondence

of UDDI, motivated this system to use the WSDL-based method of Web service discovery. The proposed system uses Bingo search engine for discovery and retrieval of WSDL of similar Web services. But due to multi-elemental nature of the WSDL it requires a huge amount of storage to store various WSDL and it also consumes high bandwidth. The processing of big WSDL increases the time of execution of WSDL processing system and thus reduces the performance but uses huge resources. Therefore the proposed system uses MOLAP for storing and processing of WSDL, thus enhancing the speed and performance.

In the proposed system, when the user gives a request for the Web services, WSDL information of semantically related web services are retrieved from Bingo search engine. WSDL of a Web service is used to describe the web services using XML form.[6] WSDL is structured and is variable in size according to a number of operations and different binding information. After retrieving WSDLs of semantically similar Web services from the online, it is very bulky to store all WSDL files in its entirety for the future processing. Therefore the proposed system uses Multidimensional Online Analytical Processing (MOLAP) to store and index only the required elements of WSDL.[7] It is also possible to process WSDL directly from the online through Document Object Model (DOM). But to compare and process similar Web services' WSDL and to invoke suitable Web services using these parameters it needs to be retrieved from the online every time when required and consumes more bandwidth of the internet. Also, it is felt that processing WSDL elements directly from the online is time-consuming when there is more traffic. Therefore the proposed system retrieves WSDL only once and stores only the required WSDL elements in the MOLAP data model. As a result, it is quickly processed instead of processing it directly from the online.

To invoke the Web services dynamically using JAX-RPC, the required parameters are needed to be populated as given in Section 3. But it is observed that in some Web services' WSDL, the name of input parameters are missing. Then being a client of the service, it is not possible to send appropriate input values to the service. Therefore it is required to find the names of unknown input parameters. In the proposed system the authors framed a rule to find out and populate the name of unknown input parameters of the WSDL by mining data set stored in the MOLAP which is discussed in Section 3.

MOLAP uses array-based multidimensional storage for a multi-dimensional view of data. The first face of the MOLAP cube stores service names, the second face stores operation names of these services. In the third face of the cube input parameters of these operations are stored.

To retrieve the information from MOLAP in detail, a deeper drill down method is used. When the user wants to know about each service in detail, the user can retrieve <service Name>, <operation name> and <input> parameters from the MOLAP using basic MOLAP operations.

To invoke the Web services dynamically, the required WSDL elements needs to be filled using MOLAP records as train data set. This approach reduces the invocation time of Web services significantly compared to the non-MOLAP model.

Due to the absence of UDDI, there is no information about the quality of Web services. Therefore the proposed system gets these QoS parameters by executing the services and stores the tested suitable services' required WSDL elements in the tested MOLAP for future use. Because of dynamic nature of Web services, it is required to refresh this tested MOLAP and the proposed system also refreshes the tested MOLAP.

Thus the objective of this research is to satisfy the user request dynamically in less time with fewer resources. The usage of MOLAP is an innovative approach to reduce the resource consumption and to increase the performance of the system. The proposed system is designed and implemented by considering the dynamic nature of the Web services.[8] Thus it is required to retrieve WSDL of the Web services at least once for each request.

The implementation of invoking of the Web Services dynamically by filling required parameters from MOLAP is given in Section 3. Section 4 gives a comparison of the results of MOLAP with the non-MOLAP model. Section 5 focuses on analysis and Section 6 the conclusion.

2. Existing Methods

The detailed literature survey provides the information about the existing Web service discovery system and the MOLAP system. In the ontology-based method, annotation has been made with reference to a domain Ontology through the standard WSDL extension mechanism.[9] Ontology-based methods provide a 'semantically enriched' version of WSDL files in order to automate complicated tasks such as service composition.

A problem in Ontologies is that they concentrate on the description of static information and do not facilitate the description of services executed by a web service except those standardized services which are part of Ontology. This contradicts the demand for a flexible description of innovative Web Services in the dynamic nature of eBusiness.

Universal Description, Discovery, and Integration (UDDI) can focus only on single search conditions such as business name, business category, business location, service type by name, discovery URL and business identifier.[10] IBM modularized this discovery engine in 2001. UDDI was a core Web service standard. It gives access to Web Services Description Language (WSDL) documents. It interprets SOAP request message. Message formats which are required to interact with the Web services were listed in UDDI. Web Services Management Software is the source of raw management data about the behavior of a web service and these are likely to be sources of QoS information available to the UDDI registry. The "QoSInformation" key value in tModel of UDDI also provides QoS information.

Jaccard similarity and Euclidian distance with WORDNET to increase the precision are used to find the similarity of services as a traditional measure.[11] And also, the useless information unrelated to the function of services, hampers the application which depends on these services. Hence in this research search engines are used to find the semantically similar Web service's WSDL files.

As the next evolutionary step after SAWSDL, WSMO-Lite (Web Service Modeling Ontology) has been proposed for describing Web Services semantically (Semantic Annotations for WSDL).[12] WSMO-Lite fills the SAWSDL annotations with concrete service semantic service descriptions. Another lightweight method for modeling functionality of web services is ontology. But WSMO-Lite relies on their derivation from free variables of precondition and effect and does not provide modeling of input and output parameters explicitly.

Discovery of services using linked social services on the Web discovers the services on a quality basis. This method not only improves quality of service discovery time but also the success rate by exploring services based on the global social network. Some of the successful service invocations and compositions are learned from those services past social interactions and are used to recommend those services as qualitative services. Even though it involved user interactions, it could not involve services' past interactions such as positive feedback or negative feedback.

The information based service discovery method uses Singular Vector Decomposition (SVD) and Probabilistic Latent Semantic Analysis (PLSA) for finding a semantic relation between Web services and user request. Authors of this system used PLSA derived by Bayesian Network Model. Web service discovery based on frequent access to the Web services is also an approach used in the Information Retrieval Method. The existing system uses methods such as term frequency (tf) and inverse document frequency (idf) i.e tf X idf are used to retrieve the Web services according to the user requests.

The fact of UDDI's absence has demonstrated that the "Internet-scale" public UDDI cannot be used by the huge number of Internet users. The authors proved that more than 53% of UDDI Business Registry (UBR) is invalid and 92% of web services cached by search engines are valid and active.[13] Search engines semantically compare the search terms entered by the user with the web service name, location, business, or tModel defined in Web Service Description File (WSDL) to give the searched results back.

Clustering using K-means, hierarchical cluster, agglomerative clustering, association rule mining, and predictive mining are different data mining techniques used in the Web service discovery. Clustering method groups the similar objects in the same group and dissimilar objects in the different groups. K-means clustering finds the distance of its centroid to the centroid of incoming pattern and includes it into its own cluster. Hierarchical cluster recursively combines the clusters until a single cluster is formed. Agglomerative clustering merges the cluster until a single tree is formed.

Authors have given in their paper that service will have different execution instances. How nicely a particular instance is executed is dependent on their non-functional property.[14] In the proposed research the non-functional properties such as response time, availability and execution time are recorded only after execution of Web Services because of the unavailability of UDDI to get those information.

Java Data Mining Web Services (JDMWS), which aims to define an interface for data mining Web services implemented in Java, provides three data mining services.[15]

1. The discovery of required data mining functionality
2. QoS ranking of services
3. Composability of services.

Authors semantically annotated the services to make it discoverable. But the proposed system is not using annotation system to make it discoverable. Instead, it uses Bingo search engine to discover required services.

Chen Wu and Elizabeth Chang have indicated that Public UDDI Business Registry - the primary service discovery mechanism over the Internet has been shut down permanently since January 12, 2006, due to several reasons.[16] Hence the most important public Web service discovery mechanism is missing from the Web Services Community. Therefore the proposed system is not using UDDI to search for WSDL.

In the paper reported the discovery of the atomic services that take part in the composition, are significantly facilitated by the incorporation of semantic information.[17] OWL-S Web service descriptions are transformed into a planning problem described in a standardized fashion, while semantic information is used for the enhancement of the composition process. The usage of semantic information in Web services presents an integrated approach for automated semantic web service discovery and composition through AI planning techniques. The OWL-S descriptions are converted into a planning problem using PDDL. Thus semantic information is used for discovery and composition of Web services in enhanced approach. But in the proposed approach OWL-S is not used for service discovery because of its static nature. This contradicts the demand of a flexible description of innovative web service in the dynamic nature of eBusiness. When large scale web services are available an innovative dynamic structured integration is required. Therefore the proposed system is not using Ontologies for getting semantic information.

The author in his paper has explained that consumers will search for web services with UDDI and manually access the web services that appear in the result.[18] The pitfalls are that UDDI search results provide only the specifications for registered web services and cannot express what consumers really want. Also, it is impossible to know the status of registered web services. But the Bingo search engine used for the search of the Web services searches semantically similar available Web services. The proposed system automatically searches and invokes the Web services according to the user requests.

Ontologies are used to analyze functional properties of a Web services with each neighbor relationship.[19] OWL-S Discovery tool is developed to retrieve Web services from IBM UDDI registries and precision and recall are used to analyze the results. In this system, user has to know the location of Web services repository and location of the directory of synonyms.

It is very difficult to get this information for normal users. UDDI registries are absent nowadays and also Ontologies are static in nature which requires modification in OWL-S according to the modified WSDL.

It is observed from the literature survey that existing systems have many drawbacks in their approach as also with respect to the performance and cost of the system. The dependency on WSDL to invoke the Web service has made the system to properly interpret WSDL elements. But sometimes WSDL parameters may not contain proper data regarding input parameters' names. For service invocation, it requires proper input parameters names which help in entering the input data. The lack of work in finding the unknown parameters also motivated the proposed system to frame a rule to get unknown input parameters.

3. Methodology

It is required to discover the requested Web services' WSDL according to the user request. The proposed system uses Bingo search engine to discover the WSDL of the Web services. The search engine retrieves semantically similar Web services according to the user given functional word.

3.1 Retrieving Similar Web Services

Functional words of user request are inputted to this system to discover the WSDL of the Web services and non-zero value of search results' precisions are used to find whether the functional words are available as Web services or not?[20]

The precision is calculated as

$$\text{Precision} = \frac{\text{Required Number of links}}{\text{Retrieved Number of links}} \quad (1)$$

The required number of links calculated as matched number of services with user requested functional word i.e. <service> name of the Web services compared with the user requested functional word using Match Making Algorithm (MMA). If the user requested functional word is matched exactly or partially with semantically similar <service> name, then those Web Services are called as required services. These required service's WSDL records are processed to extract WSDL's <input> element and

Table 1. Semantically related WSDL for "weather" as requested function through Bingo search engine

Requested Word	Service key Retrieved	Retrieved WSDL Links
Weather	GlobalWeather	http://webservicex.net/globalweather.asmx?wsdl
	ndfdXML	http://graphical.weather.gov/xml/SOAP_server/ndfdXMLserver.php?wsdl
	WeatherForecast	http://www.webservicex.net/WeatherForecast.asmx?WSDL
	Weather	http://wsf.cdyne.com/WeatherWEB SERVICE/Weather.asmx?wsdl

Table 2. Partial data set of MOLAP after storing the required WSDL elements

Dimension 1 Level1	Dimension 2 Level2		Dimension 3 Level3	
Service Key	**Operation key**	**Type of Binding**	**Input Parameters**	**Number of inputs**
Weather Get_Cities_By_Country Get_Weather Get_Cities_By_Country Get_Weather Get_Cities_By_Country	Get_Weather	Soap12	Parameters	2
	Soap12	Parameters	1	
	HttpPost	CountryName CityName	2	
	HttpPost	Country	1	
	HttpGet	CountryName CityName	2	
	HttpGet	Country	1	
WeatherForecast	GetWeatherbyPlaceName	Soap12	Parameters PlaceName	1
	GetWeatherbyPlaceName	HttpPost	PlaceName	1
	GetWeatherbyPlaceName	HttpGet	PlaceName	1
	GetWeatherbyZipcode	Soap12	Parameters	1
	GetWeatherbyZipcode	HttpPost	Zipcode	1
	GetWeatherbyZipcode	HttpGet	Zipcode	1

then it is compared with input requirement of the user request.

For example, if the word "weather" is inputted as a functional word then the proposed system prepares the query in the form of Functional_word? WSDL i.e. "weather? WSDL" to the Bingo search engine.

The result of the search query is WSDL links and other similar additional details. From these additional details, only WSDL links are extracted. Extracted WSDL links for "weather" as functional word and for the query "weather? WSDL" is given in Table 1.

3.2 Storing WSDL Elements in MOLAP

When the search query is given with "weather" as a functional word, it resulted into four WSDL links according to the Bingo search technique. Similarly, for other different requests, it resulted in a different number of links. When the system downloads the contents of these WSDL links every time, it consumes the internet and also to process WSDL parameters it is required to be stored in the file. Because of its big size, it cannot store all WSDLs in separate files. The similar Web services' WSDL can be written on the same WSDL file as the WSDLs are processed. But for future analysis, it is required to download these WSDL once again which consumes additional internet.

Therefore the proposed system uses MOLAP cube to store required WSDL elements of similar Web services as different data sets (in different rows of MOLAP) and uses for future analysis. At the initial stage, MOLAP cube is used to store the training data of WSDL elements of retrieved Web Services according to the user-requested functional word.

The first face/level of MOLAP gives similar Web services' <service> elements which can be accessed using one dimension data structure. The second face/level shows <operation> elements of these services which will be accessed using two dimension data structure. The third face / level stores <input> parameters of these operations which will be accessed using three dimension data structure. These WSDL elements in Table 2 become the training data set which requires mining technique to fill unknown input parameters.

3.3 Mining MOLAP Data Set

It is observed in the data set that <operation> elements of some WSDL of Table 2 contain "parameters" or "Body" or "unknown" in the place of the name of the parameter of the <input> element of WSDL. These parameters are unknown and the name of this unknown parameter is to be mined to get required parameter's name.

To do this a rule is written according to the elements of the same operation with different binding protocols. The logic for writing the rule is as follows.

For given WSDL records of raw data set of particular <operation> element

If <service>/<operation>/<Input> having "parameter" or "Body" as the data then

If same <service>/<operation> available with HttpPost or HttpGet binding with known parameter then populate <service>/<operation>/<Input> element of that <operation> in the place of unknown <input>

According to the given train data set it is observed that <service>/<operation>/<Input> of SOAP binding is stored as "parameter" or "Body" in the Table 2. This is taken as unknown_parameter.

If <service>/<operation>/<Input> of same operation of HttpGet or HttpPost binding contain real parameter name then it is taken as known_parameter here.

$$(<service>/<operation>/<Input> = \text{"parameter"} \text{ or } <service>/<operation>/<Input> = \text{"Body"}) \rightarrow \exists(<service>/<operation>/<binding> = \text{"HttpPost"} \text{ or } <service>/<operation>/<binding> = \text{"HttpGet"}) \text{ and } <service>/<operation>/<Input> = known\_parameter\ (S) \quad (2)$$

$$\text{Support(s) of Antecedent} = \frac{\text{count(Antecedent)}}{\text{count(Records having the same operation)}} \quad (3)$$

$$\text{Support(s) of Consequent} = \frac{\text{count(Consequent)}}{\text{count(Records having the same operation)}} \quad (4)$$

From the Table 2 of training data set of "Weather" <service> for "GetWeather" <operation> Support of Equation 2 is = 1/3 = 0.33.

From the Table 2 of training data set of "Weather" <service> for "GetWeather" <operation> support of Equation 3 is = 2/3 = 0.67.

For different Web services, as seen in table 2, it results in different support value. If support>0 for Equation 3 and Equation 4 then unknown parameter is replaced by a known Parameter.

3.4 Multi-Dimensional OLAP (MOLAP) Operations

Different MOLAP commands are implemented in the proposed system using suitable data structure. These are used for retrieving the WSDL elements in different element levels. These commands are given here.

- **Roll-up:** Climbing up a concept hierarchy for the dimension reduction. When the user tries to roll-up from operation to service names "Roll-up operation name" command is used.

In this system, Roll-up Advanced CheckAddress gives AddressLookup service name.

Roll-up GetWeatherbyZipcode gives WeatherForecast service name.

- **Drill Down:** Stepping down a concept hierarchy from the dimension of services to the dimension of operations. This navigates to highly detailed data from less detailed data. This is required when a request is given to see service operations of a particular service. Now the service name is drilled down to see its operations.

Drill Down command to "AddressLookup" service gives these operation names: <StandardizedAddress, AdvancedStandardization, ReturnCityState, BarcodeFontURL, CCServerCheck, CheckAddress, CheckAddressW2lines AdvancedCheckAddress, VersionInformation, CalculateDistanceInMiles, AlternateCities, CityStateToZipCodeMatcher, GetCongressionalDistrictByZip> in three sets for three different types of binding protocols i.e. Soap1.2, HttpGet and HttpPost protocols.

- **Slice:** It is given when subcube is to be generated for a given cube. For example, when the user is willing to get input parameters of the particular operation then "slice operation name" command is used.

In this system slice operation = "AdvancedCheckAddress" gives AddressLine2, StateAbbrev, AddressLine, ZipCode, City, LicenseKey in three sets for different types of binding. Similarly for all other operations slice command gives their input parameter names.

- **Dice:** Dice selects two or more dimensions from a given cube and provides a new sub-cube.

For example dice for service_name=" AddressLookup" and operation=" CheckAddress" results in three sub-cubes which are explained below.

First sub-cube gives the service names equal to "AddressLookup", operation name equal to "CheckAddress" and input= StateAbbrev, AddressLine, ZipCode, City, LicenseKey for Soap1.2 binding.

Second sub-cube gives the service names equal to "AddressLookup", operation name equal to "CheckAddress" and input= StateAbbrev, AddressLine, ZipCode, City, LicenseKey for HttpGet binding.

Third sub-cube gives the service names equal to "AddressLookup", operation name equal to "CheckAddress" and input= StateAbbrev, AddressLine, ZipCode, City, LicenseKey for HttpPost binding.

3.5 Web Service Invocation

The proposed system dynamically invokes the Web services by extracting and filling the required parameters from MOLAP cube. The invocation of Web services involves filling the required parameters as given from Step 1 to Step 8 with JAX-RPC statements.

(i) It is required to get WSDL link of required web services from MOLAP (Table 2) after mining the unknown input parameters.

In the MOLAP cube, first face stores service information. The WSDL link will be retrieved using this service information with the first dimension of the MOLAP as given in the statement given below.

wsdllocation= serviceInformation[cnt].Location;

(ii) It is necessary to store the namespace information of the same service from MOLAP(Table 2)

String namespace = serviceInformation[cnt].NameSpace;

(iii) To get operation key of same Web service.

The operation element is stored in the second face of MOLAP which is retrieved as

opName=operation_keylevel2[cnt][count];

(iv) It is required to create service object by using the above WSDL URL, namespace and service name

Service service = (Service) factory.createService(new URL(wsdllocation),new QName(namespace,service_keylevel1[cnt]));

(v) To specify type of binding that is SOAP binding QNAME method is used with namespace and service name as parameters.

QName portName = new QName(namespace,service_keylevel1[cnt]+"Soap");

(vi) To specify the operation name to be invoked once again QNAME method with namespace and operation name is used as given here.

QName operationName = new QName(namespace,opName);

(vii) To create call method using operation and port information

Call call = (Call)service.createCall(portName, operationName);

(viii) To get information about the number of the input elements of operation from third face of the MOLAP(Table 2) it is required to loop through the number of inputs. In the following code snippet inputlevel3[cnt][count][j] indicates input parameters of required service and required operation.

for(int j=0;j<operationInformation[cnt][count].NumofInputs;j++)

parameters[j]=inputlevel3[cnt][count][j];

(ix) Invoke the web services using already prepared call method.

Result of webServices=call.invoke(parameters);

Step 8 gives the results of invoked Web services. As observed from Step 1 to Step 8 of the logic, service_keylevel1[cnt], operation_keylevel2[cnt][count], inputlevel3[cnt][count][j] are extracted from the MOLAP cube to fill the service name, operation name and input parameters dynamically to invoke the web services. The MOLAP mining has already replaced unknown input parameters with known parameters. Hence there will not be any trouble in invoking the Web services. Thus trained MOLAP data gives proper parameters according to the given functional word of the requester.

If the Web service is invoked successfully then these WSDL elements are stored in another MOLAP of the same structure. This is called tested MOLAP data set. This tested MOLAP data set is used to get information of tested Web services.

4. Results

The proposed system is implemented using eclipse 64bit IDE and tested at constant Internet speed and recorded the difference of time in the invocation of web services using MOLAP and without using MOLAP as specified in the above-given sections.

The time required to invoke web services using MOLAP is less than the time required to invoke Web services without MOLAP and it is justified during the experiment. The result of the implementation is shown in Figure 1.

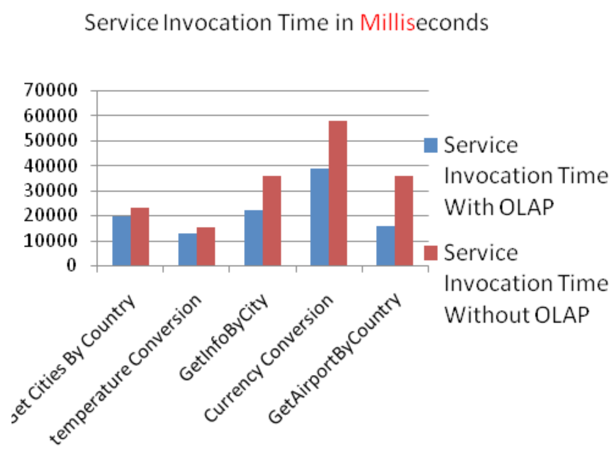

Figure 1. Comparison of Service Invocation time of MOLAP & Non-MOLAP.

The difference in time justifies the advantage of the usage of MOLAP. The time required to invoke the WEB Services using MOLAP is less due to two reasons. The first is, it is not required to download WSDL from the internet to extract required WSDL elements that are to be filled dynamically. The second is, it is not required to open WSDL files to read and extract required elements. The MOLAP stored the required extracted elements of WSDL by which it could be possible to fill the required parameters for invocation of Web services as given in Section 5. The system without MOLAP requires downloading the parameters from the internet for service invocation. Therefore this will take more time comparing to MOLAP.

The proposed system also saves the time during storing the WSDL elements in the MOLAP because instead of using DOM directly from the browser, this system stores the retrieved WSDL in the file once. Then the required elements of this WSDL file are stored in the MOLAP. Other retrieved WSDL are overwritten in the same WSDL file to save the storage space. The search technique of the Bingo search engine retrieves the links of the WSDL of the similar "Zipcode" services for the request "ZipCode?WSDL" is shown in Table 3.

Table 4 shows the WSDL links retrieved for the request "Temperature?WSDL" from the Bingo search engine. Similarly for other requests similar WSDL files retrieved indicate there are similar Web services and if any one of the Web services failed then other similar Web services are used. These similar WSDLs' required elements are stored in the same MOLAP as different from which required rows are extracted as and when needed.

As shown in the Table 5 the average search speeds for different Web service's search system are different for different types of users. The time to search the Web services is more for general users because they may not know the proper input values to be given to execute. The IT professionals may also be slower than Web service professionals because the service professionals have proper knowledge of the usage of Web services. As shown in the comparison table, the speed of the proposed system is high compared to other systems.

All the approaches except the proposed system are dependent upon UDDI, which is absent nowadays. The proposed system does not depend on UDDI and it depends only on search engine to retrieve WSDL. Hence proposed system is more effective than the other existing systems, is justified.

Table 3. Retrieved similar WSDL links for "ZipCode?WSDL" from Bingo search engine

ZipCode	USZip	http://www.webservicex.net/uszip.asmx?WSDL
	AddressLookup	http://Web Service.cdyne.com/psaddress/addresslookup.asmx?wsdl
	ZipCode	http://www.ripedev.com/webservices/ZipCode.asmx?WSDL

Table 4. Similar WSDL links retrieved for "Temperature?WSDL" from Bingo search engine

GlobalWeather	http://www.webservicex.com/globalweather.asmx?WSDL
TemperatureConversions	http://webservices.daehosting.com/services/TemperatureConversions.wso?WSDL
WeatherForecast	http://www.webservicex.net/WeatherForecast.asmx?WSDL
CurrencyConvertor	http://www.webservicex.net/CurrencyConvertor.asmx?WSDL

Table 5. Speed Comparison of the different Web service Search Systems

Approaches	General Users	IT Professionals	WebService Professionals
Seekda.com	1100 s	721 s	650 s
ProgrammableWeb	1950 s	1205 s	980 s
Iserve	1020 s	605 s	550 s
OWLS-MX	950 s	520 s	495 s
SOAS	310 s	295 s	205 s
The Proposed System	60 s	45 s	35 s

5. Analysis

Among many Web service search systems, a specific system called Woogle acquires Web service descriptions through UDDI registries.[21] In this system, the search is performed through keywords and offers the possibility to discriminate between inputs and outputs. The proposed system does not depend on UDDI because of its absence. In the proposed system the WSDL link of the Web services is retrieved through the Bingo search engine.

The result is shown in Table 4 for the request "temperature?WSDL". Similar Web services links hold "currencyConversion.WSDL" which do not belong to the temperature category. Therefore this system compares user requested functional word with every operation elements of the retrieved WSDL links and also necessitates the computation of WSDL weight matrix for selection of a suitable operation to satisfy the user requests.[22]

VUSE is another example of specialized web service search engine that retrieves the WSDL.[23] VUSE also depends on UDDI to retrieve WSDL and it retrieved a total of eight WSDL links for "weather" as a search query. But now a days some of the WSDL links among those links do not contain the services. Therefore the proposed system tests the services by invoking them and stores the WSDL of only the tested services in the tested MOLAP.

This tested MOLAP justifies the availability of services and used for future requests. But due to dynamic nature of the Web services, tested MOLAP cannot be used for a long time and it is required to refresh the WSDL elements of the MOLAP. The refreshing procedure involves retrieving the WSDL from the online and storing only the required elements as given in Section 3.

The raw data set of MOLAP is mined to store the known parameters in the place of unknown <input> parameters' name. Mining is possible only if the WSDL contain known <input> parameter in atleast any one of the binding types among HttpPost or HttpGet or SOAP binding. If there are no such types of binding then names of input parameters can be found by using the operation names. This system also implements the interpretation of <input> element by using "By", "To" and "From" clauses of <operation> name element, only if <input> parameter is not interpretable using the framed rule of Equation 2.

Authors in the paper analyzed QoS attributes like response time, price, and security in different faces of OLAP cube. They stored quality of service parameter in the multi-dimensional data model.[24] In this model, they stored security, response time and price of the Web services in different faces of the web services. The QoS parameters are extracted from the service details of UDDI repository. But in the proposed model there is no way to get QoS parameters before execution of the Web services. Therefore authors stored service name, operation name and input parameters in the MOLAP cube. After execution of the Web services, the QoS parameters are computed and used for future requests.

The Table 5 shows the performance metrics in terms of average search speed of results for different Web service search systems.[25]

As shown in the Table 5 the average search speeds of different Web service's search system are different for different types of users. The time to search the Web services is more for general users because they may not know the proper input values to be given to execute. The IT professionals may also be slower than Web service professionals because the service professionals have proper knowledge of the usage of Web services. As shown in the comparison Table 5, the speed of the proposed system is high compared to other systems.

It is observed in the results that implementation of the proposed system using MOLAP is an innovative approach which saves the system resources and satisfies the customer by returning the results quickly.

6. Conclusion

The proposed system used Bingo search engine to get semantically similar Web services according to the user requested functional words. The retrieved WSDL of the web services are interpreted to get their functionality and to find matching operations to the user requested functional word. During WSDL processing, the unknown input parameters are resolved using an association rule. To overcome the problem of consumption of huge storage space, the WSDL elements are stored in the MOLAP cube. From the results it is justified that usage of MOLAP cube reduces the consumption of internet and increases the performance of the system.

Due to the absence of UDDI, it is not possible to get information about the quality of Web services. Therefore the proposed system gets these QoS parameters by executing the services and stores the tested suitable services' required WSDL elements in the tested MOLAP which can be used in the future. Because of dynamic nature of the Web services, it is required to refresh this tested MOLAP and as a future work a system can be developed which refreshes the contents of the tested MOLAP.

The user requested functions may be available either as single web service or as composed Web services. As a future work, it is planned to link tested composable web services' WSDL parameters using MOLAP cube.

7. Acknowledgements

I thank my guide Dr. Niranjan N. Chiplunkar for his great support and motivation in the implementation of this research work. I also like to thank late Dr. Ashok Kumar, for his help and blessings at the beginning of my research work. I also thank all the reviewers, editors-in-chief for their great support in reviewing this paper.

8. References

1. Web Services Tutorial. Date accessed: 15/3/2014. Available from: http://www.tutorialspoint.com/Webservices
2. Simple Object Access Protocol. Date accessed: 25/6/2013. Available from: http://www.tutorialspoint.com/soap/what_is_soap.htm
3. OWL-S: Semantic Markup for Web Services. Date accessed: 18/4/2014. Available from: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122
4. Tan PN, Steinbach M, Kumar V. Introduction to data mining. Addison-Wesley. University of Minnesota; 2013. p. 490–526.
5. Bose A. Effective Web service Discovery using a Combination of Semantic Model and Data Mining Technique [Master's Thesis]. Queensland University of Technology, Brisbane, Queenland, Australia; 2008. p. 22–35.
6. Search Engine. Date accessed: 03/02/2014. Available from: http://www.bingo.com
7. Data Warehousing – OLAP. Date accessed: 06/01/2015. Available from: http://www.tutorialspoint.com/dwh/dwh_olap.htm
8. Pawar S, Chiplunkar NN. Necessity of dynamic composition plan for web services. Proceedings of first International Conference on Applied and Theoretical Computing and Communication Technology (ICATCCT); 2015; Davangere. p. 737–42.
9. OWL-S: Semantic Markup for Web Services. Date accessed: 08/04/2015. Available from: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122
10. Universal Description Discovery and Integration. Date accessed: 10/01/2013. Available from: https://en.wikipedia.org/wiki/ Universal_Description_Discovery_and_Integration
11. Liu F, Shi Y, Yu J, Wang T, Wu J. Measuring Similarity of Web Services Based on WSDL. Proceedings of IEEE International Conference on Web Services; 2010; Miami, FL. p. 155–62. Crossref
12. Thomas F, Johaness R, Friedrich S. Towards Knowledge Discovery in the Semantic Web. Proceedings of International Conference MIKWI; 2010; Jena. p. 1151–62.
13. Khalid E, Ahmed E, Hassan, Patrick M. Clustering WSDL Documents to Bootstrap the Discovery of Web Services. Proceedings of IEEE International Conference on Web Services; 2010; USA. p. 150–8.
14. Shiting W, Chaogang T, Qing L. Probabilistic top-K dominating services composition with uncertain QoS. International Journal of Service Oriented Computing and Applications. 2014 Jun; 6(10):91–103.
15. Janciak I, Brezany P. A Reference Model for Data Mining Web Services. Proceedings of IEEE Sixth International Conference on Semantics, Knowledge and Grids; 2010; Beijing. p. 251–8. Crossref
16. Wu C, Chang E. Searching Services on the Web: A Public Web Services Discovery Approach. Proceedings of Third International IEEE Conference on Signal-Image Technologies and Internet-Based System; 2007; Shanghai. p. 321–8. Crossref
17. Hatzi O, Vrakas D, Nikolaidou M, Bassiliades N, Anagnostopoulos D, Vlahavas I. An Integrated Approach to Automated Semantic Web Service Composition through Planning. IEEE Transactions on Services Computing. 2012; 5(3):319–32. Crossref
18. Noh-sam P, Lee G. Agent-based Web services middleware. Proceedings of IEEE Global Telecommunications Conference (GLOBECOM). 2003; 6:3186–90. Crossref
19. Amorim R, Claro D.B, Lopes D, Albers P, Andrade A. Improving Web Service Discovery by a Functional and

Structural Approach. Proceedings of IEEE International Conference on Web Services; 2011; Washington DC. p. 411–8. Crossref

20. Pawar S, Chiplunkar NN, Kumar A. Dynamic Discovery of Web Services. International Journal of Information Technology and Computer Science. 2014; 6(10):56–62. Crossref
21. Dong X, Halevy A, Madhavan J, Nemes E, Zhang J. Similarity search for web services. Proceedings of VLDB; 2004. p. 372–83. Crossref
22. Pawar S, Chiplunkar NN. Populating Parameters of Web Services by Automatic Composition Using Search Precision and WSDL Weight Matrix. International Journal of Computational Sciences & Engineering. Forthcoming 2016.
23. Hatzi O, Batistatos G, Nikolaidou M, Anagnostopoulos D. A Specialized Search Engine for Web Service Discovery. Proceedings of IEEE 19th International Conference on Web Services; 2012; Honolulu, HI, USA. p. 448–55. Crossref
24. Song J, Hou H, Tiantian L, Guoqi L, Zhiliang Z. QoS Cube: Management and Navigating Web Services through Multi-dimensional Model College of Software. Proceedings of the 14th IEEE International Conference on Computational Science and Engineering; 2011; China. p. 9–15. Crossref
25. Chen, Wuhui, Incheon P. Improving efficiency of service discovery using Linked data-based service publication. International Journal of Information System. Springer Publishers. 2013 Sep; 15(4):613–25.