AN IMPROVED WEB SERVICE DISCOVERY FRAMEWORK BASED ON SIMILARITY MEASUREMENT AND QOS RANKING

R. Jeberson Retna Raj.[1], Dr.T.Sasipraba.[2]

[1]Research Scholar, Sathyabama University, Chennai, India
[2]Professor & Dean, Sathyabama University, Chennai, India,
Email: [1]jebersonin@yahoo.co.in

ABSTRACT

Reducing the irrelevant web services and not missing the most relevant web services are the two complimentary issues in web service discovery process. Due to the lack of support from Universal Business Registries (UBR) for complex queries, web service search engine has been introduced to discover the pertinent web services. Traditionally, web services can be discovered by keywords. Since the keywords are unable to bear the underlying the semantics of the search query, operation based discovery has been introduced. Furthermore, clients search interest often changed, we propose a model to discover similar web services based on QoS requirements. To achieve this, find the degree of the functional component that matches with the search query can be computed, and its respective QoS attributes that can be normalized by the proposed algorithm. These two factors can be aggregated to provide overall score value for the web service. For a search request, the list of candidate services will be provided to the client based on the degree of closeness with the search query. The system has been tested with the real and synthetic data, and the test result shows very favorable results.

Key words: web services, QoS attributes, Normalization, web service discovery.

I. INTRODUCTION

Proliferation of web services paves a way for many service providers implement their business services using web services. Omnipresent of the internet and technology provides incentive to the service providers to create, locate and access required web services regardless of platform. Since the web service becomes the realization of Service Oriented Architecture (SOA), developers relying on this technology for its support on reusability and interoperability [1]. Due to the increasing number of web services in the internet for similar functionality, identifying the pertinent web service to the client's requirement is a challenging research problem in services computing paradigm [2]. Due to the vigorous proliferation of web services, locating the desired web service for a given request has become a very valuable task [3]. Due to the availability of a large number of web service providers, it is often the case that a large number of services are available for a given request. The selected service must incorporate all the requested operation. Reducing irrelevant services and not missing the most relevant services are the two complementary issues about the selection process. Current research shows most of the services implemented by. NET web service. For a request, more than hundred service providers readily available in the internet to satisfy the request. Due to the failure of the public registries, search engines are the alternate tool to discover web services. Generally, web services can be searched by keywords [4]. If the searched query does not contain in the part of web service name exactly, then the service may not be retrieved. [5] Furthermore, the user not able to specify their search request more precisely rather than by keywords [6]. Traditional web search engines devoted to website and not for web services as they are unable to articulate the search query well [10]. As the semantics of website and web service are differs, we argue that a dedicated web service search engine to discover the pertinent web service. Furthermore, a client want to search a web service, current systems are unable to deliver the requested task accurately. So we advocate that the system assist the user to discover web services based on QoS requirements. Our contribution is in two fold, first to store all published web services in the service pool. Extract the metadata such as service, portType, message, operation elements. Compute the quality of web services for response time, throughput, availability and reliability using the metrics. Secondly, compute the functional and non functional evaluation by the proposed algorithm. Aggregating the functional and non

functional values to provide the rank value so that the client can get the web service based on QoS requirements. The gist of this approach is the consumer can specify the search request as input operation and its expected output operation. Furthermore, the client can specify the required QoS attributes. Based on the request, a list of candidate services which are matched with the request will be provided to the client for setting weights over the QoS attributes. Since this search is based on input and output operations, the unwanted services will be filtered from the list. So how wisely select the appropriate web services for the benefit of the customer is a key issue to the selection process.

The paper is organized as follows. The brief introduction of web service discovery and the related work done in the proposed technique is discussed in section II. Section III describes the proposed QoS driven web service discovery and ranking process for efficient web service selection. Section IV discusses with the experimental results of a search query based on keyword search as well as operation based search. Finally, section V end with conclusion.

II. WEB SERVICE DISCOVERY

Locating the desired web service is an important research issue in services computing paradigm. Due to the advent of the internet and proliferation of web services which makes internet as a huge resource library for a client to locate, select and compose business applications. Furthermore, competition among web services is ready to satisfy a client request. A business process consists of participating an individual or group of web services to execute a task. Service discovery is an integral part of semantic web services. Since traditional methods don't support the Quality of search, the need of the hour is a system which assists the user to select the pertinent web service for a client request. Selection of web service becomes challenge when the composition of the web services is to be done automatically.

A. Limitation of UDDI and search engines

Universal Business Registry (UBR) terminates their registry operations from early 2006 for public use. Due to the failure of the UDDI, the clients are forced to use an alternate search method such as search engines and crawlers etc. During the discovery process in UDDI, a client may experience several setbacks. Most of the UDDI are seldom updated and significant parts of information in these registries are out of date [18]. Due to the lack of semantics descriptions of web services, the results returned by the registries are effectively inadequate. Universal Business Registries (UBR) such as Microsoft, SAP, IBM are close their registry operations for public use. The reason behind their shut down is cited as the goal is achieved. The entries of the web services are scaled up day by day basis so that duplication of entries may inevitable. Generally, UDDI is based on keywords [4]. The consumer has to do view-select-request queries several times. These enormous hardships may hindrance the consumer to locate the desired web service [5]. Moreover, once the desired web service is not available in the internet, the search process again restarted. Recent survey shows more than 53% of the UBR registered services are invalid. This kind of procedure leads to less usage of UDDI and relying on web service pool. Searching web services using search engines is also based on keyword which is inherently impractical for discovery process. Search engines do not understand the web service functionality outlined in the description file [16].

The idea behind the keyword based search is that the keyword involved in the query which matches them with web service description. Since the keyword based searching unable to match the underlying semantics of web service, they may miss the relevant results and returns irrelevant one to the client. User unable describes the search request more precisely than keyword is another limitation of keyword search. Furthermore, keywords do not suffice for accurately specifying user's information needs. Web services are developed and maintained by their provider. Suppose the service is modified or no longer available, the service consumer have to repeat the discovery process.

B. Operation Based Search

A web service is a reusable software component interacts over systems in a heterogeneous environment. Web services can be discovered, located and invoked across the internet. A web service consists of interface description and implementation description used to expose the service functionality. Interface description containing the type and definition elements and implementation definition containing port type, binding and service elements. A web service consists of several operations. Each operation deals with individual task assign to them for execution. The

operation contains an asset of parameters. These parameters are structured in a hierarchy by using complex types. The operation based search is to provide the search result more accurately than the keyword based search. Since the web service operation is going to be as part of the application, the user would like to prefer operation based search than the keyword based search.

C. Web service discovery classification

Current discovery approaches can be classified into two categories namely syntactic discovery and semantic discovery. Syntactic discovery incorporated the techniques such as UDDI based search, text document based search, schema matching and software component matching. Semantic-based discovery is mainly based on ontology. Generally, UDDI is based on keyword search. Seldom updating of the service providers information and simultaneous access will not be encouraged are the serious limitation of UDDI based search. Typically, text document search is widely used in search engines and web pages [6]. This is also based on keywords. Search engines are unable to articulate and bear the underlying semantics of the searched query so that search engines are inherently impractical for web service discovery. The functionality of schema matching is based on to capture clues about the semantics of schema and suggests matches based on them [7][8]. Operations are typically much more loosely related to each other than are tables in a schema and each web service in isolation has much less information than a schema. Finally, software component matching has defines the problem of examining signature (data type) and specification (behavior) matching [9]. It is mainly for software reuse. For Semantic discovery, Ontology based discovery is one of the approach for discovering web services [5]. The ontology based discovery approach suffers from performance problems due to the use of ontology reasoners. Furthermore, constructing ontology as a semantic backbone for a large number of distributed web service is really not easy. These are the major setback for ontology based discovery. All the aforementioned approaches don't deal with QoS based operation search. Thus the proposed QoS based web service discovery stress the need of searching a web service based on QoS based input/output operation.

D. Overview of the proposed Service Discovery

Proliferation of web services is expected to introduce competition among large number of web services that offer similar functionality. Selecting highly configurable web services in terms of performance is of paramount important for web service selection. Since the keyword search unable to provide the accurate search result for a client request, operation based search has been introduced. In this approach, the single similar and composite operation which incorporates with the QoS requirements can be searched. The idea is to retrieve the metadata of input/output operation from a WSDL and apply a mining algorithm used to measure the co-occurrence of terms and cluster the terms into a set of concepts, and leverage these concepts to determine the similarity of inputs/outputs operations.

Assume all the web services are published. The web service crawler collect the list of web services from the internet and stored in the service pool. The WSDL parser extracts the metadata from the WSDL such as service name, operation name, input, output type, message and it can be can be stored in the service pool. QoS attributes of a web service is normalized by the proposed algorithm and the appropriate match score value is stored in the service pool. Whenever a request has been performed, QoS consultant selects a list of pertinent candidate web services that are matched with the input/output operation will be provided to the client.

E. Related Work

QoS based web service discovery is an important solution for filtering and selecting between functionally equivalent web services stored in registries or other repositories. Unfortunately, Functional discovery of web service is not adequate and may circumvent the client interest, as there may be hundreds of equivalent web services. Users further must be assisted in selecting the appropriate web service for their needs. Recently, several works has been done in QoS based web service discovery. Xin Dong Et al., proposed "Similarity Search for Web Services approach for web services discovery" [24] for searching web services. They construct a search engine named as woogle used to search similar services. The clustering module used to group the similar web service operation and generate a meaningful concept. However, the work is not supported the QoS based search for a user request.

Xuanzhe Liu [6] proposed a model for homogeneous web service discovery. The search model is based on input/output operation. Web service operations are extracted from WSDL file and stored in the service pool. The Single similar and composite operations are grouped based on terms and leverage these terms into concepts by the clustering algorithm. The Composite operations can be grouped by agglomeration graph algorithm. Furthermore, the service provider changes of web service contents can be updated by atom feed. In[17], a web service search engine WSExpress has been introduced. The functional value and the non functional values are aggregated to get the overall score value used to index the web services for a search query. Authors in [23] proposed URBE(Uddi Registry By Example) for web service retrieval model. The model is based on bipartite graph clustering the related services for retrieval process. However, the model doesn't incorporate with the client QoS requirements.

III. PROPOSED QUALITY DRIVEN WEB SERVICE DISCOVERY

QoS deals with a set of nonfunctional properties of a web service that encompasses performance characteristics. As the user more concern about the performance of web service they use, QoS can be used to discriminate functionally similar web services. Discovering a web service according to the client requirement is a complex task as many services are available to satisfy a request. Search a service using keyword which circumvents the user to get an appropriate web service instead of offering a less specific one. Moreover, irrelevant information may be provided to the client which leads to take a vague decision as which one will be the most suitable one. To alleviate these difficulties, QoS driven web service discovery has been introduced. The gist of this approach is to provide the required web service based on users QoS requirements. Figure 1. Shows the architecture of QoS based web service discovery process.

The QoS driven web service discovery model is to search a web service according to the client requirement. The proposed system assists the client to search web services based on input/output operations. The web services of single similar and composite operations can be searched based on user's QoS requirements. The system allows the user to specify the input operation and the output operation incorporate with the QoS requirements. The functional details are used to identify the operations and the QoS requirements will be normalized to calculate the merit of non functional requirements. Matchmaking algorithm used to match the client request with the available services.

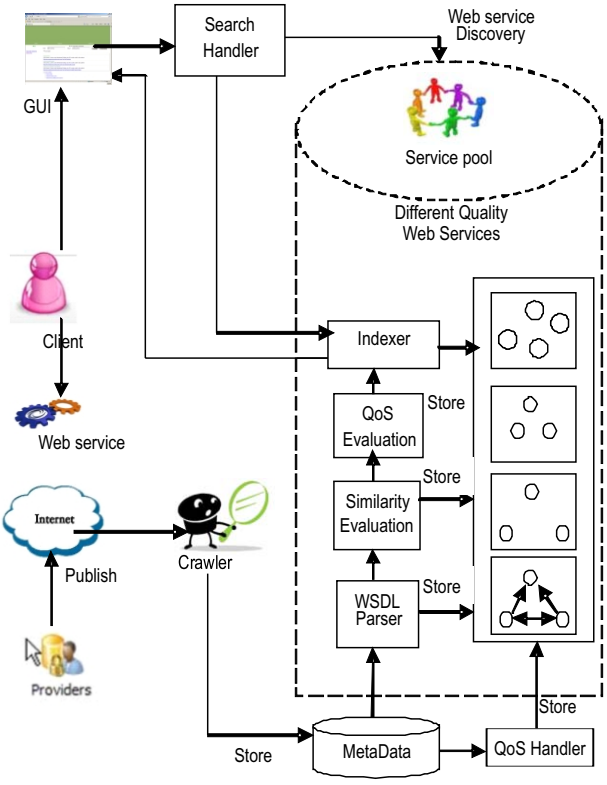

Fig. 1. Proposed QoS driven web service discovery architecture

The architecture of the proposed QoS driven web service discovery has been shown in Figure 1. The web services descriptions are searched by a crawler and stored the WSDL in the database. The QoS handler stores all the QoS attribute values and using the normalization algorithm to normalize the values and store it in service pool. The WSDL parser extracts the Meta data from a WSDL file and stores it in service pool. The cluster agent cluster the terms and generate the concept which will be stored in a service pool. The feed manager used to index the web services. Whenever a client performs a search via user interface, the consultant selects the pertinent web services to the client's QoS requirements. The Service pool manager consists of all discovery components including similarity

cluster. The proposed architecture consists of web service Meta data extraction and clustering module which extracts the input/output operation from WSDL file and store these Meta data into the service pool. The extracting features from WSDL documents will be done in a offline process. The QWS Dataset [21] [22] consist of 2507 web services WSDL files are taken for experiment, the input and output operations are extracted from the service descriptions. The extraction process used in [14] can be followed for extract the metadata such as web service name, type, ports, operations, input/output name and stored in a service pool.

Web Service Metadata extraction and Clustering

The Meta data requirement of a web service is shown in table I, the Meta data can be extracted from WSDL file and stored in the service pool. For example, Table II shows the Meta data of currency converter web service and its equivalent location is shown in Table III. The idea is to identify the terms and cluster the terms into concepts, and leverage these concepts to determine the similarity of inputs/outputs operations.

F. Service Meta Data Model

A web service can be defined in terms of Meta data as follows.

WS = {Name, Message Exchange, Data types, set of operations}}

Where operation = {no. of operation, input, output}

The operations can be described by name and text description in WSDL defined by port Type element and associated with atleast one input message and one output message.

Each input and output contains a set of parameter for an operation defined by the message element and type element used in the message.

A message has optionally m parts m>=1(atleat one)

A simple wsdl structure for simple operation is shown in figure 2.

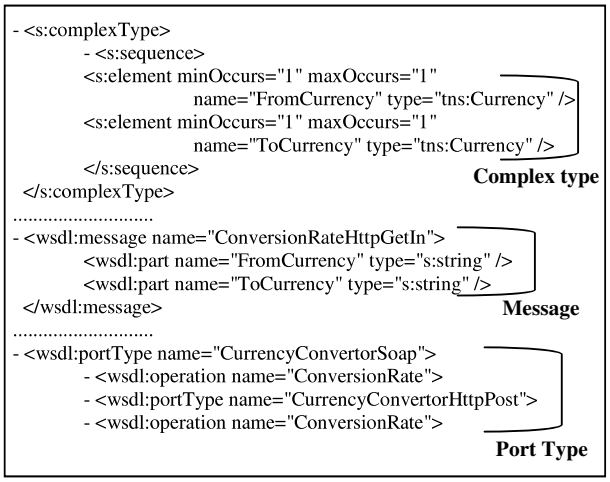

Fig. 2. Excerpts from Currency converter web service WSDL

Service pool

The structure of the service pool is represented as the vector of SP={Service Name, Profile, Op, Input/output, list of QoS} Table 1 shows the tag entries which are used in service pool.

Table 1. Naming rules of Service Pool

Tag	Naming Rules
Service Name	The service name of the web service
Profile	The domain and functionality description of the service providers in the pool
Op	The operation name of the web service
Input	The input name can be used for client search
Output	The output name can be used for client search
QoS values	List of QoS attributes

Table 2. Example of Generating a Pool WSDL

	Name	Input	Output	operation
WS_1	Currency Rates	get Currencies List	getRate	get Currencies List
WS_2	DOTS Currency Exchange	Convert From Currency	Convert To Currency	ConvertCurrency
WS_3	Currency Rates	From Currency Code	To Currency Code	get Conversion
WS_4	Currency Convertor	From Currency	To Currency	Conversion Rate
WS_5	Currency Server Web Service	src Currency	dst Currency	get Currency Value

Table 2 described about 5 currency converter web services with similar functionality. The metadata of service pool consist of name, input, output, and operations, which are derived from WSDL file. The WSDL file definition, message, port Type, binding and service elements. To search a web service based on operation, we require the extracted features of name of the service, input/output name, and operation included in port type element and stored in service pool. Table 3 shows the 25 different web services retrieved by the crawler.

Table 3. Functionally Similar Web Services - Currency converter

Currency converter Web service URLs -WSDL (25 web services)
http://ws.strikeiron.com/HouseofDev/currencyrates?WSDL
http://ws2.serviceobjects.net/ce/CurrencyExchange.asmx?WSDL
http://ws.strikeiron.com/HouseofDev/currencyrates151?WSDL
http://www.webservicex.net/CurrencyConvertor.asmx?wsdl
http://www.currencyserver.de/webservice/currencyserverwebservice.asmx?WSDL
http://www.webservicex.com/CurrencyConvertor.asmx?wsdl
http://currencyconverter.kowabunga.net/converter.asmx?WSDL
http://glkev.webs.innerhost.com/glkev_ws/Currencyws.asmx?wsdl
http://www.currencyserver.de/webservice/currencyserverwebservice.asmx?WSDL
http://allysoft.ru/BScurrency/currency.asmx?WSDL
http://fx.cloanto.com/webservices/CurrencyServer.asmx?wsdl
http://www.atlaz.net/webservices/GetCurrencyExchange.wsdl
http://server1.pointwsp.net/ws/finance/currency.asmx?WSDL
http://www.freewebs.com/jimmy cheng/CurrencyExchangeService.wsdl
http://ws.soatrader.com/gama-system.com/1.0/CurrencyExchangeRates?wsdl
http://tvazteca.viajez.com/WServicesDev/CurrencyRequest?WSDL
http://ws.serviceobjects.com/ce/CurrencyExchange.asmx?WSDL
http://ws2.serviceobjects.net/ce/CurrencyExchange.asmx?WSDL
http://www.petermeinl.de/CurrencyConverter/CurrencyConverter.asmx?wsdl
http://currency.niekutis.net/currency.asmx?wsdl
http://cs.daenet.de/webservice/CurrencyServerWebService.asmx?WSDL
http://www.xignite.com/xCurrencies.asmx?wsdl
http://trial.serviceobjects.com/ce/CurrencyExchange.asmx?WSDL
http://ws.strikeiron.com/ForeignExchangeRate3?WSDL
http://ws.soatrader.com/baydonhill.com/0.1/Currency?wsdl

G. Concept clustering

To measure the similar and co-occurrences of term can be identified using association rules. Obtaining the association rules for the term set, we then try to cluster the concept set. Agglomerating algorithm [6], this is used to generate the concept from terms. i.e. Input/output, data types. The over frequent terms and infrequent terms are filtered in this step.

H. Predicting the Similarity

The similarity measurement of web service operation can be achieved by employing traditional TF/IDF (Term Frequency/Inverse Document Frequency) measurement. A web service operation op can be represented by three tuple vector. For a given two operations we can find the similarity by combining the similarity of each individual elements respectively.

1. By using the traditional TF/IDF measurement, estimate the text description of operation and the web services the operation belong to.
2. Estimate the similarity of input and output by considering the underlying semantics the input/output parameter cover.

   Input $=<$ min, CI Where min - text description of input names and concepts that associates with

   Similarly the output Output $=<$ nout, Co $>$
3. Find similarity of input in the following two ways.

   Evaluate the similarity of the description of input names by TF/IDF.

   Split min into a set of terms. We should filter the terms related to outputs (eg. “ZipCode” in the input “City Name ByZipCode”). Then we replace each term with its corresponding concepts, and then use the TF/IDF measure.

The output can be processed in a similar fashion. Now, we define the similarity between two operations op_i, op_j the following formula

$$\text{Sim}\ (op_i, op_j) = {}_1\text{Sim}(N_{wi}, N_{wj}) + w_2\text{Sim}(N_{\text{opi}}, N_{\text{opj}})$$

$$+ w_3\text{Sim}\ (\text{input}_i, \text{input}_j) + w_4\text{Sim}(\text{output}_i, \text{output}_j)$$

Here, $wi(i = 1, 2, 3, 4)$ is the weight assigned to similarity of operation text description, input and output, $\Sigma\, wi = 1$. Then, we define the two operations that are similar as follows:

Given two operations op_i, op_j a threshold w_1 then claim that op_i, op_j, are similar operations if $Sim(op_i, op_j) \geq w_i$

I. Constructing Service Aggregation Graph

Two service discovery types are specified against a given request of input and output. We are follow the same approach used in [6] and the step is as follows,

1. A list of similar single operations that can accept the request and

2. A sequence of operations that can be composed to fulfill the request.

In our approach, we process these two requirements within a directed graph , $G=<V, E>$, in which:

1. V is the vertex set. Each vertex in the graph G corresponds to an operation in our data set;

2. E is the edge set. Each edge corresponds to the input or output of the vertex.

To find single similar and composible operation in the web service pool, a "Service Aggregation Graph" (SAG) algorithm can be used [6].

J. QoS based Web Service discovery process

Generally, the UDDI and search engines are used to search the services by keyword. Since the keyword search is not able to extract the underlying semantics of the searched query, operation based search has been introduced. The discovery process completed only if the QoS based requirement incorporate with operation based search. Thus QoS driven web service discovery approach efficiently selects the required web service. The system assists the user to specify the input operation and the output operation along with the QoS requirements. QoS values of the web service such as response time in millisecond, throughput in seconds, cost in dollar, both reliability and availability in percentage. These heterogeneous values such as millisecond, second and percentage can be normalized by the simple weighted average method. The range of the normalized value is between [0, 1].

We have taken four metric for QoS assessment. First metric measures the average execution time of a web service, it takes the values from the integer set [1, ∞], while its measurement unit in milliseconds. The second metric measures the average throughput of a web service, its values also belong to the integer set [1, ∞], whereas its measurement unit in seconds. The next two metric measures the average availability and reliability of a web service, it takes the real values and is computed as percentage. Table 4 consist of QoS web service parameters which are used in [19] such as response time, throughput, reliability and availability that can be retained in the proposed method.

Table 4. QoS Attributes for web service selection

Parameter	Definition	Unit of measurement
Response time	Time taken to send a request and receive a response	Millisecond
Throughput	Total number of invocations for a given period of time	Invocations per second
Reliability	Ratio of the number of error messages to total messages	Percent
Availability	Number of successful invocations/total invocations	Percent

Let S be the set of preselected services, describes the utility score decreased if the QoS value is increased. Lower is better policy can be maintained. Here, the response time, throughput and cost are come under this category. $QOS^{increment}$ represents the positive QoS attribute such as availability and reliability. If the QoS values are increased then the utility value also in a increased direction. Higher is better policy is taken under this category.

$$s \in S \; if \; \forall_i = 1.....N \begin{cases} if \; Q_i \in QOS^{Decrement} \; (q_i)_r \leq q_i(s) \\ if \; Q_i \in QOS^{Increment} \; (q_i)_r \geq q_i(s) \end{cases}$$

To illustrate matchmaking step, consider five instance of web service WS1, WS2, WS3, WS4, WS5 with respective offers: these QoS values are normalized by equation (2)

$$WS_1 \; (QoS) = \{ 205.33ms, 3.5s, 80, 60 \}$$

$$WS_2 \; (QoS) = \{ 104.75ms, 19.4s, 89, 73 \}$$

$$WS_3 \; (QoS) = \{ 126.25ms, 3.6s, 100, 60 \}$$

$WS_4\ (QoS) = \{\ 648\text{ms},\ 0.6s,\ 92,\ 73\ \}$

$WS_5\ (QoS) = \{\ 539.29\text{ms},\ 6.2s,\ 63,\ 80\ \}$

K. Normalization

At this stage all the eligible services offer a quality level that is equal to or higher than the requested and come at the affordable costs. We will thus evaluate service offers in terms of the gain in quality and cost that is proposed. Let Q and C be the evaluation metrics of gains in quality and cost respectively. We here by define q as scalar values between 0 and 1.

We first evaluate the gain in each quality dimension. For each, we define two parameters $(q)_{\max}$ and $(q)_{\min}$ as follows

$$\begin{cases} (q)_{\max} = \begin{cases} \max_{s \in S}\ q_i(s) & if Q_i \in QoS^{Increment} \\ (q_i)_r & if Q_i \in QoS^{Decrement} \end{cases} \\ (q)_{\min} = \begin{cases} \min_{s \in S}\ q_i(s) & if Q_i \in QoS^{Increment} \\ (q_i)_r & if Q_i \in QoS^{Decrement} \end{cases} \end{cases} \quad (1)$$

The scaling function is defined as $Utility_i(q_i)$ and takes values in [0, 1]. $Utility_i(q_i)$ is increasing for $QoS^{increment}$ attributes and decreasing for $QoS^{Decrement}$ attributes.

$$Utility_i(q_i) = \begin{cases} \dfrac{q_i - (q_i)_{\min}}{(q_i)_{\max} - (q_i)_{\min}} & if Q_i \in QoS^{Increment} \\ \dfrac{(q_i)_{\max} - q_i}{(q_i)_{\max} - (q_i)_{\min}} & if Q_i \in QoS^{Decrement} \\ 1 & if\ (q_i)_{\max} - (q_i)_{\min} \end{cases} \quad (2)$$

We can easily show that for all $i = 1 \ldots N$, $Utility_i((q_i)_r) = 0$. We now derive the scalar metric Q from the vector $(Utility_i(q_i))$, $W = (w_1), (w_2), \ldots$ denotes consumer's quality preferences. Where $0 \le W \le 1$ and $\Sigma_{k=1}^{N}\ W_i = 1$.

The rank value of the web service can be calculated as

$$Rank = \sum_{i}^{n}\ Utility_i(q_i) * W_i \quad \ldots (3)$$

IV. EXPERIMENTAL RESULTS

Discovering a web service according to the client requirement is an elusive task as similar services are ready to satisfy a request. To alleviate the difficulties, QoS driven web service discovery has been introduced. The important elements which are participated in the message exchanges are service consumer, service provider, service pool manager, and the feed manager. The Feed manager used for subscribing the Service to the client. The system user interface allows the service provider to register their services. For example, Webservice.net, it successfully added it to the service providers list. The WSDL parser extracts the Meta data such as name of web service, input/output operation, web service operation from WSDL file. For example, as stated in table 1 for currency converter web service, name of the web service is "Currencyconverter", input message operation as "Fromcurrency", output message operation as "Tocurrency" and the operation name as "ConversionRate". These Meta data are extracted by the WSDL parser and stores it into the Service pool.

Whenever a request comes, the consultant communicates the service pool manager for selecting a suitable web service. The service pool manager finds the single similar and composite operations by using the clustering algorithm and agglomeration graph algorithm. Based on this computation, the related concepts are grouped and the list is provided to the client for setting weights over QoS attributes.

L. Keyword search

As service providers publish the web service content on the internet using standard WSDL interface, people from anywhere can access the web service on any platform and any operating system. The system interface allows the user to specify the keyword and in response the system returns the list of matched service to the client. For example, the keyword "currency" , the services which are matched with the searched query currency will be returned. Here all the 25 web services are returned as a search result.

The search engines are searched by using a plain textual description, which describes the general kind of service that is offered, for example, service related to "weather forecasting" or "travel agency." As the keyword search not able to provide the underlying semantics of the web service search operations, we

advocate the homogeneous service discovery mechanism for searching a web service.

M. Input/output operation based search

Service Pool contains the Meta data information of the WSDL file. It's not only maintains the operations with similar inputs/outputs, but also the "hyperlink" between the operations. This approach combines multiple sources of evidence to determine the similarity. The search domain of the operation that captures the purposed functionality such as "GetWeatherByZipCode," "SearchBook," or "QueryAirplaneTimetable." Finally, we find the data type deriving from the input/output. The data types do not relate to the low-level encoding issues such as integer or string, but to the semantic meanings such as "weather," "zipcode," etc. Service Pool can return two types of results: the operations with similar inputs and outputs, and the sequentially composible operations. The component Service Pool analyzes each WSDL file, filter the irrelevant information, and retrieve terms from the input/output data types.

The service consumers can search the homogeneous Web services within their Web browser. The users can type in their desired input and output names and get the returned results. It shows the similar single operations and composite operations into two parts. The similar operations are highlighted as well as their inputs and outputs, in the description of the Web services they belong to. The composite operations are listed in a tree, where the upper node's output can be accepted as input by the lower ones. The returned results are sorted by their similarity against the requests. And the user can able to subscribe the service.

Figure 4 shows the search for similar services based on input and output operations. So the system will return the single similar operation against the given input and output operations. For example, getting similar services, the request contains input operation GetCurrency and respected output parameter GetCountries. The search has been performed and the list contains the requested web services.

The following figure 3 shows the user interface for searching composible operations web services. Composible in the sense the output of the service operation will be the input of the others in a chain manner. This type of functionality has been tested by the sample search data as Get Currency and requested output operation is GetCountries. By getting this data a search has been conducted and the composible operaration based services has been listed and which is shown in figure 6. The resulted service operation agiaint the given request is GetCountries, GetCurreuncyByCountry, GetCurrencyCodeByCurrencyName. This output has been shown in a tree structure format.

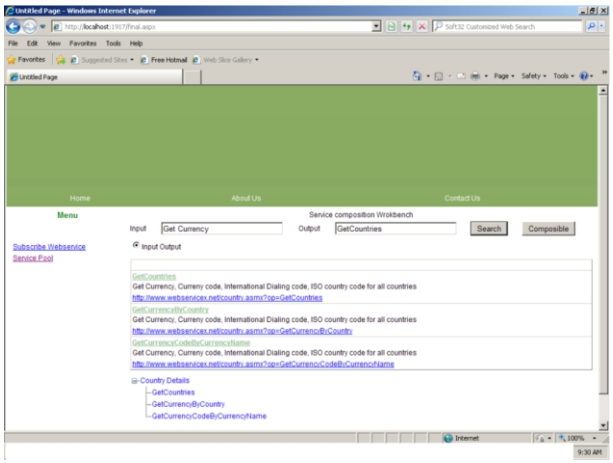

Fig. 3. Searching Composible operations

For specifying the composite search operation Get Currency, the GetCountries composite web service has been open for invocation. By invoking GetCountries web service another web service GetCurrencyByCountry has been opened. By specifying India as a country name the details of the currency details will be provide in a XML description by calling GetCurrencyCodeByCurrencyName web service as this invocation in a chain and interrelated, this kind of search service is composite search is rightly justified.

Subscribing the Service

In ServicePool, the Feed Manager component attaches the metadata to the Atom feeds and indexes the feeds to the corresponding Web services by using the standard Atom publish protocol. Here, it is necessary to make the binding between the Atom feeds and WSDL, to ensure the consistency with the original Atom syntax and semantics. Due to the functionalities that Service Pool provides, we adopt two feed types. The first one is the Web service entry, which is used to subscribe exactly one Web service. This element represents Web service by binding the metadata of the Web service to an Atom feed. Figure 4 shows

subscribing web service to the client. The system has been tested with the subscription of data.

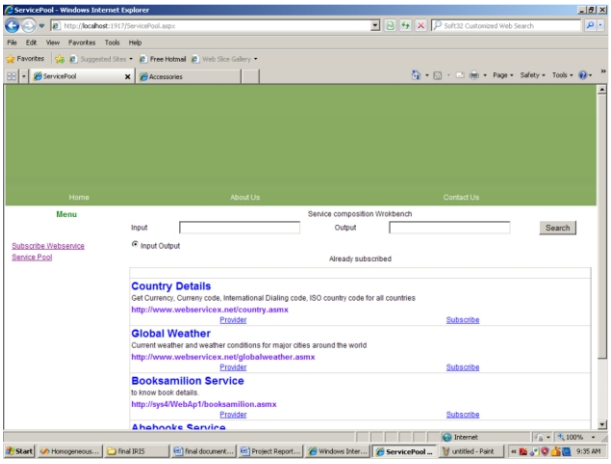

Fig. 4. Subscribing a web service using RSS Feed

QOS based discovery

Here the system is tested for searching the service based on Quality of service parameters. The parameters considered here is Response time, Throughput and Availability and Reliability for selecting optimal web service. The QWS dataset can be used to validate the proposed system [18][20]. These QoS values can be measured by standard benchmark tools [21][22].

Table 5. The QOS values for 5 Functionally Similar Web Services

Service	Response time (Milli Second)	Throughput (Second)	Availability (Percentage %)	Reliability (percentage %)
WS1	205.33	3.5	90	60
WS2	104.75	19.4	89	73
WS3	126.25	3.6	100	60
WS4	648	0.6	92	73
WS5	539.29	6.2	63	80

Thus 5 different services of CurrencyConverter web service are registered with varying QoS Values. Table 5 shows the QoS requirement of a currencyconverter web service.

A user want to search the web service by input/output operation name means, they can use the exclusively designed interface along with QoS requirements. The system allows the user to specify the web service by input/output operation and QoS attribute he is looking for can be made through the user interface as a search query. The QoS attributes such as response time, throughput, reliability and availability are normalized and a match score value will be identified and it will be compared to the available web services. The highest matched services will be provided to the client for setting preference over QoS attribute. Table 6 show the normalized value getting from equation (2).

Table 6. Normalized QOS Values

Service	Response time	Throughput	Availability	Reliability
WS1	0.81485	0.84574	0.72972	0
WS2	1	0	0.70270	0.65
WS3	0.96042	0.84042	1	0
WS4	0	1	0.78378	0.65
WS5	0.20011	0.70212	0	1

The client has set weightage for all the QoS parameters varying on its importance to him [0,1]. Based on these weightages the rank value of the services is evaluated using equation (3).

If the user assigned weight to each QoS attribute as {0.8,0.7,0.7,0.8}, the utility value can be represented by

$$(0.8\ 0.7\ 0.7\ 0.8)\begin{pmatrix} 0.8148 & 0.8457 & 0.7297 & 0 \\ 1 & 0 & 0.70270 & 0.65 \\ 0.9604 & 0.8404 & 1 & 0 \\ 0 & 1 & 0.7837 & 0.65 \\ 0.2001 & 0.7021 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0.6518 & 0.5929 & 0.5107 & 0 \\ 0.8 & 0 & 0.4918 & 0.52 \\ 0.7683 & 0.5882 & 0.7 & 0 \\ 0 & 0.7 & 0.5485 & 0.52 \\ 0.1600 & 0.4914 & 0 & 0.8 \end{pmatrix} = \begin{pmatrix} 0.4388 \\ 0.4529 \\ 0.5141 \\ 0.4421 \\ 0.3628 \end{pmatrix}$$

Table 7. Rank Value for Web Service Using Proposed Ranking Algorithm

Service	Rank value
WS3	0.5141
WS2	0.4529
WS4	0.4421
WS1	0.4388
WS5	0.3628

The result is evaluated by weighted sum of all normalized QoS parameters and the final rank value is evaluated. According to this the rank value will be provided to the client from higher to lower order. So the recommendation will be based on higher is better can be followed.

A. Performance Analysis

In general, Web service discovery is based on keywords. Due to this search the user can get relevant as well as irrelevant result also. In the proposed system, the search is based on input and output search So that the consumer can get the relevant web services.

The following table 8 shows the comparison between keyword based search and operation based search mechanisms.

Table 8. Comparison between Keyword search and operation based search

Search Operation choice	Keyword	Input & Output
Keyword : Weather	2	1
Input & output : weather conditions (i/p) Get weather (o/p)		
Keyword : Currency	8	4
Input & output : Currency details (i/p) Get cities by country (o/p)		
Keyword : Books	7	4
Input & output : Author (i/p) Get Books (o/p)		

The Following figure 5 shows the Comparison Chart for Keyword Search and Input/output based Search

V. CONCLUSION

Reducing irrelevant services and not missing the most relevant services are the two complementary issues about the selection process. These issues are addressed and a recommendation framework for web service discovery is elaborately presented in this paper. The single similar operation and composite operations are identified by the proposed method. Furthermore, the similar terms are clustered and the concepts are

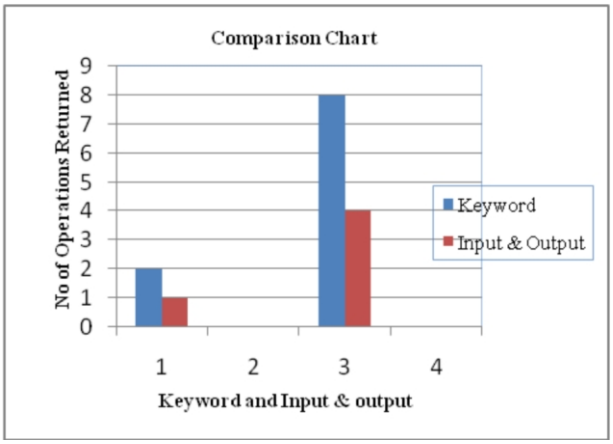

Fig. 5. Graph shows the difference between keyword search and input/output operation based search for weather and currency converter web service.

obtained by clustering algorithm. The QoS computations of web services are measured and using normalization algorithm which provides rank values. Whenever a search has been performed, the system selects the list of matched services from the service pool and provided to the client. The test bed result shows that a sharp deviation between the keyword based search and operation based search. The proposed method incorporates with the client's non functional requirement which shows a promising result to the searched query. In future, the agglomerative algorithm can be replaced by hybrid hierarchical clustering algorithm to reduce the incorrectly grouped of object at an early stage can be alleviated.

REFERENCES

[1] Joyce El Haddad, Maude Manouvrier, and Marta Rukoz "TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition", IEEE Transactions on Services Computing, Vol. 3, No. 1, January-March 2010, pp.73-85

[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] Dimitrios Skoutas, Dimitris Sacharidis, Alkis Simitsis, and Timos Sellis, "Ranking and Clustering Web Services Using Multicriteria Dominance Relationships", IEEE Transactions on Services Computing, Vol. 3, No.34, July-September 2009

[4] M. Brian Blake, "Knowledge Discovery in Services", IEEE Internet Computing, March/April 2009. pp.88-91.

[5] M. Brian Blake, "Knowledge Discovery in Services (KDS):Aggregating Software Services to Provide Complementary Data", IEEE Transactions On Knowledge And Data Engineering

[6] Xuanzhe Liu, GangHuang, "Discovering Homogeneous Web Service Community in the User-Centric Web Environment", ", IEEE Transactions on Services Computing, Vol. 2, No. 2, April-June 2009

[7] E. Rahm and P.A. Bernstein, "A Survey on Approaches to Automatic Schema Matching," VLDB J., vol. 10, no. 4, pp. 334-350, 2001.

[8] H.H. Do and E. Rahm, "COMA: A System for Flexible Combination of Schema Matching Approaches," Proc. Very Large DataBase Conf. (VLDB '02), 2002.

[9] A.M. Zaremski and J.M. Wing, "Specification Matching of Software Components," ACM Trans. Software Eng. and Methodology, vol. 6, pp. 333-369, 1997.

[10] Al-Masri, E.; Mahmoud, O.H.; Discovering Web Services in Search Engines, IEEE Internet Computing, May-June 2008 74 - 77

[11] Almeida, J.; Gonçalves, M.A.; Figueiredo, F.; Pinto, H.; Belem, F.; on the quality of information for web 2.0 services, IEEE Internet Computing, 2010 Issue:6 November-December 2010, PP:47-55

[12] Aliaksandr Birukou, "Improving Web service discovery with usage data", IEEE software,November/December 2007,PP:47-54

[13] George Zheng and Athman Bouguettaya, "Service Mining on the Web", IEEE Transaction on Services Computing, Vol2, No.1, January-March 2009. PP:65-78

[14] Khalid Elgazzar, "Clustering WSDL Documents to Bootstrap the Discovery of Web Services", 2010 IEEE International Conference on Web Services, PP:147-154

[15] Fangfang Liu "Measuring Similarity of Web Services Based on WSDL", 2010 IEEE International Conference on Web Services, PP:155-162

[16] Yin Baocai et.al, "A Framework and QoS Based Web Services Discovery",IEEE International Conference on Software Engineering and Service Sciences (ICSESS), 2010 , PP:755-758.

[17] Yilei Zhang, Zibin Zheng, and Michael R. Lyu, "WSExpress: A QoS-Aware Search Engine for Web Services", 2010 IEEE International Conference on Web Services, PP:91-98

[18] Eyhab Al-Masri and Qusay H. Mahmoud, "Toward Quality-Driven Web Service Discovery", IT Pro May/June 2008, PP:24-28

[19] Richi Nayak Bryan Lee "Web Service Discovery with additional Semantics and Clustering", 2007 IEEE/WIC/ACM International Conference on Web Intelligence, PP:555-558

[20] Al-Masri, E., and Mahmoud, Q. H., "Discovering the best web service", (poster) 16th International Conference on World Wide Web (WWW), 2007, pp. 1257-1258.

[21] Al-Masri, E., and Mahmoud, Q. H., "QoS-based Discovery and Ranking of Web Services", IEEE 16th International Conference on Computer Communications and Networks (ICCCN), 2007, pp. 529-534.

[22] Eyhab Al-Masri and Qusay H. Mahmoud , "Investigating Web Services on the World Wide Web", ACM, *WWW 2008,* April 21–25, 2008, Beijing, China, PP:795-804.

[23] Pierluigi Plebani and Barbara Pernici "URBE: Web Service Retrieval Based on Similarity Evaluation" IEEE Transactions on Knowledge And Data Engineering, Vol. 21, No. 11, November 2009, PP:1629-1642.

[24] Xin Dong Alon Halevy Jayant Madhavan Ema Nemes Jun Zhang, "Similarity search for web services" VLDB '04 Proceedings of the Thirtieth international conference on Very large data bases - Volume 30

R.Jeberson Retna Raj is a research scholar from Sathyabama University, Chennai. Currently He is doing his Ph.D in Computer Science and Engineering in the area of Web Services. His research interest includes Service oriented architecture (SOA), web services, GIS web services etc. He had presented various papers in national & international Journals and conferences