A Future Image for Web Services' Discovery with a Client Web based Interface

Amal Yousief  
Department of Systems and Computers Engineering,  
Faculty of Engineering,  
Mansoura University,  
Mansoura, Egypt University,  
Mansoura Egypt

Dr. Hesham Arafat  
Department of Systems and Computers Engineering  
Faculty of Engineering  
Mansoura University,  
Mansoura, Egypt University,  
Mansoura Egypt

Dr. Ahmed Saleh  
Department of Systems and Computers Engineering  
Faculty of Engineering  
Mansoura University,  
Mansoura, Egypt University,  
Mansoura Egypt

ABSTRACT
Web Services’ discovery is a very important issue related to Web Services. From Syntax and using match-making words to semantic web and taking QoS parameters into account for selecting between Web Services having the same functionality for finding the best service that fulfills the customer’s requirements. In this paper we put a future image for Web Services’ Discovery by merging both UDDI and Search Engines, as the new trend in Web Services’ Discovery is building a central repository storing all Web Services after collecting them from UDDI, different UBRs …, this central point will be a reference to the client for searching the required Web Service. Two datasets one contains 365 WS and the other 2500 WS used in the experimental work. Our work will cover two phases from the suggested model phase 4 and phase 6 fig.(3).Classifying Web Services before storing them will enhance the search process and it could be a step for building open web directory contain all Web Services like open web directory includes a USDL description document for each service which can be provided in a combination of several qualities or properties of a service, the discovery problem involves finding a specific service that can fulfill the given input and output criteria in the query based on a syntactical equivalence of the input and output names. In Semantic Discovery We assume that a directory of services has already been compiled, and that this directory includes a USDL description document for each service. A USDL description of the desired service can be written, a query processor can then search the service directory for a “matching” service Quality of Service, or QoS, is “a combination of several qualities or properties of a service” and helps us to select a proper Web-service from the web applications.

1. INTRODUCTION
The service discovery process can be divided into two phases: the first one is based on the functional aspects of the service (i.e. input and output parameters, what the service does, preconditions) while the second one deals with non-functional parameters (i.e. QoS). Web Service, the magic word which will give a bright future for communications through the internet, Web Services are hardware, programming language, and operating system independent, it will move us from hardware distributed systems to software distributed systems. It is the newest branch in web engineering, we can differ between service and Web Service in a few words service is a provision of value in some domain (i.e. when somebody wants to book a flight ticket for travelling from Riyadh to Cairo it doesn’t matter whether the requester goes to an airline tickets office or uses the airline Web site to book his flight) fig.(1), where Web Service is a computational entity accessible over the Internet.

![Fig 1 : Web Service](image-url)
It is a set of non-functional attributes that may influence the quality of the service provided by a Web service like Availability, Capacity, Reliability, Performance, Cost, Response Time. There are two major problems in dynamic Web Services’ discovery with QoS. The first involves the specification of QoS information. How should the QoS information be expressed and/or stored? A standard format must be agreed upon and used in order for the information to be exchanged and interpreted. The second problem is one of matching the customer’s requirements with that of the provider. For example, if a customer is looking for services that matches its QoS requirements of 2ms response time, 400Kbps throughput and 99.9% availability, how can services be found whose QoS advertisement satisfies these requirements? [13]. Our work is considered with the second problem. Collecting Web Services data is not the key element that leads to an effective Web Services’ discovery, but how it is stored. The fact that Web Services data is spread all over existing search engines databases, accessible UBRs, or file sharing platforms does not mean that clients are able to find these Web Services without difficulties. However, making this Web services data available from a standard, universal access point that is capable of aggregating this data from various sources and providing clients to execute search queries tailored to their requirements via a search engine, so merging between both UDDI and Search Engines might be a good alternative to using UBRs for Web Service discovery, particularly when considering information accuracy. Search Engines are trends for finding and discovering Web Services also emerged in recent years. Search Engines such as Google, Yahoo, AlltheWeb and Baidu have become a new source for finding Web services. [20] this new trend is due to limitations found with services registries like UDDI [23]. Search Engines have become a new major source for searching for Web Services. Yet, they are vulnerable to returning irrelevant results and only provide access points to WSDL documents while UDDI business registries provide a more business-centric model that can be used as the first step towards an application-centric Web, so merging between both UDDI and Search Engines in the process of Web Services discovery is the future trend. The UDDI registry can be supported with external database, which stores non-functional information about Web Services. Ran S. proposed a model by introducing a Web Service QoS Certifier module [26]. Gang YE, Chanle WU, Jun YUE, Shi CHENG introduced A QoS-aware Model for Web Services’ discovery by suggested a new UDDI registry which is a repository of registered Web Services with lookup facilities[27], the proposed new registry differs from the current UDDI model by having information about the functional description of the Web Service as well as its associated QoS registered in the repository, but their model does not modify the standard UDDI interface and the client side. Hailhua Li, Xiaoyong Du, Xuan Tian [28] constructed a SAM-based service model for Web Services management, SAM-WS, and proposed an approach for users to explicitly describe their QoS requirements using QoS ontology. Eyhab Al-Masri and Qusay H. Mahmoud [29] proposed a novel exploration engine, the Web Service Crawler Engine (WSCE). WSCE is capable of crawling multiple UBRs, and enables for the establishment of a centralized Web Services’ repository which can be used for large-scale discovery of Web Services they didn’t cover the client side too. The proposed by Julian Day system [30] consists of two parts: augmented client and the QoS forums. Clients send their experiences to a central Web Service which stores this information inside an internal database. This Web Service can be thought of as a kind of forum system for QoS information. It can respond to requests about particular Web Services, sending all the data it knows about a particular service to a requesting client. Now when a client wants to pick a service, he/she gathers information from the QoS forums, and then reasons about

for Web Services’ discovery depending on Search Engines, for solving the problems with Web Services’ discovery and using UDDI a tModel[14] was suggested for storing values of QoS for each Web Service. A reputation-enhanced [16] model where service matching, ranking and selection algorithm is presented and evaluated. In spite of newer technologies, service registries still provide the foundation for cataloging and classifying Web Services. The UDDI Business Registry (UBR) is the central service directory for publishing technical information about Web Services, but the existing UDDI specification has some major technical limitations that make it an incomplete solution for Web Service discovery [17]. Due to UDDI Limitations which are(i) UDDI wasn’t intended to serve as a search engine for Web Service discovery,(ii) UDDI registration is voluntary and thus can easily become passive. (iii) UDDI doesn’t provide any guarantee of the validity or quality of information it contains. (iv) A disconnection exists between UDDI and the current Web. UDDI is incapable of providing quality-of-service (QoS) measurements for registered Web Services. UDDI doesn’t maintain or provide any Web Service life-cycle management. So, Search Engines might be a good alternative to using UBRs for Web Service discovery, particularly when considering information accuracy. Search Engines are trends for finding and discovering Web Services also emerged in recent years. Search Engines such as Google, Yahoo, AlltheWeb and Baidu have become a new source for finding Web services. [20] this new trend is due to limitations found with services registries like UDDI [23]. Search Engines have become a new major source for searching for Web Services. Yet, they are vulnerable to returning irrelevant results and only provide access points to WSDL documents while UDDI business registries provide a more business-centric model that can be used as the first step towards an application-centric Web, so merging between both UDDI and Search Engines in the process of Web Services discovery is the future trend. The UDDI registry can be supported with external database, which stores non-functional information about Web Services. Ran S. proposed a model by introducing a Web Service QoS Certifier module [26]. Gang YE, Chanle WU, Jun YUE, Shi CHENG introduced A QoS-aware Model for Web Services’ discovery by suggested a new UDDI registry which is a repository of registered Web Services with lookup facilities [27], the proposed new registry differs from the current UDDI model by having information about the functional description of the Web Service as well as its associated QoS registered in the repository, but their model does not modify the standard UDDI interface and the client side. Hailhua Li, Xiaoyong Du, Xuan Tian [28] constructed a SAM-based service model for Web Services management, SAM-WS, and proposed an approach for users to explicitly describe their QoS requirements using QoS ontology. Eyhab Al-Masri and Qusay H. Mahmoud [29] proposed a novel exploration engine, the Web Service Crawler Engine (WSCE). WSCE is capable of crawling multiple UBRs, and enables for the establishment of a centralized Web Services’ repository which can be used for large-scale discovery of Web Services they didn’t cover the client side too. The proposed by Julian Day system [30] consists of two parts: augmented client and the QoS forums. Clients send their experiences to a central Web Service which stores this information inside an internal database. This Web Service can be thought of as a kind of forum system for QoS information. It can respond to requests about particular Web Services, sending all the data it knows about a particular service to a requesting client. Now when a client wants to pick a service, he/she gathers information from the QoS forums, and then reasons about

Fig 2: A programmable Explorer Bar for Microsoft’s Internet Explorer Web Browser [33]

2. RELATED WORK
Many studies, approaches and ontologies manipulated Web Services’ Discovery with respect to QoS. The most of the proposed frameworks and architectures use UDDI registry for storing Web Service descriptions. When Web Services were in hundreds all approaches for Web Services’ discovery were concerned with UDDI and UBRs Registries when number of Web Services increased and become in thousands a new trend
which service is best. Authors at[31] suggested an ontology named DAML-QoS; it is a complementary ontology that provides detailed QoS information for DAML-S (Darpa Agent Markup Language for Services) users. A programmable Explorer Bar for Microsoft’s Internet Explorer Web Browser[33] that uses Common Sense Reasoning to display contextually relevant tasks based on what the user is viewing, and allow users to find and directly query Web Services. The Web Services Explorer Bar contains two areas Search Web Services and Tasks. The Search Web Services area allows users to query SOAP based Web Services using natural language. The Tasks area displays contextually relevant tasks based on what Web page the user is viewing.

3. PROPOSED ENHANCED TECHNIQUE FOR WEB SERVICES’ DISCOVERY

Collecting Web Services data is not the key element that leads to an effective Web Services’ discovery, but how it is stored. Using data collected by crawlers our work could be summarized into three steps (1)Classifying Web Services due to URL or a function, (2)Storing Web Services information both functional and non functional(QoS),(3)building a client web based ( GUI )for Web Services’ discovery.

Fig.(3) represents a future image of Web Services’ discovery process based on QoS by merging UDDI and Search Engines for enhancing the discovery process through six phases our research is concerned with phase four and six , phase4 is a WS-Classifier which classified the collected Web Services in phases one , two and three(crawlers) before storing them in the Web Services Storage, phase6 is a Web Service Storage Search Engine based on QoS which will be a GUI to the client for getting the best Web Service which he/she searches for. Classifying Web Services after fetching their WSDL documents and storing them will enhance Web Service discovery as Web Services with the same function grouped together after that the best Web Service which fulfill consumer requirements will be selected with respect to QoS (Quality of Services Parameters) according to URL or function. A new technique for managing the storage process of WSS.After classifying Web Services was implemented, the structure of the database that store data of Web Services like Name, Function, WSDL and QoS values must be changed to a new one for matching the decision tree classifier (Decision trees are powerful and popular tools for classification and prediction).

The new technique will help in (i)Crawlers’ function Support , using the new technique for WSS Management will support the function of crawlers , storing web services’ information and metadata will be easier also help updating Web Services’ information like QoS (ii)WSSSE-QoS’ function Support .WSSSE-QoS (Web Service Storage Search Engine based on QoS) is a system that will be established as an intermediate between WSS and consumer used for selecting the best web service that will fulfill the consumer requirements due to QoS, the new technique for WSS Management will support the operation of WSSSE-QoS, which will facilitate the Find and Select operations and make them easier which will enhance web service discovery,(iii)WSS Stability ,the new technique of WSS Management will affect stability of WSS because the new infrastructure could be implemented using data structure (using trees) as storing data inside WSS through the indexing module IM. IM is primarily responsible for building data structures over textual information contained within WSDL interfaces or UDDI objects(i.e. business Entity, business Service,binding Template, (Models, among others)[22].(v)Ability of Large Extension The new infrastructure of WSS will be able for large Extension, because tree structure has this advantage and this will be suitable for the future growth of web services numbers in the future. For implementing the new technique discussed above web directories are more suitable for storing web services data.Decision Trees was the most suitable algorithm for the suggested technique, which is a classifier in the form of a tree structure .Decision trees are powerful and popular tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural networks, decision trees represent rules. Rules can readily be expressed so that humans can understand them or even directly used in a database access language like SQL so that records falling into a particular category may be retrieved fig.(4).
Taking into account phase 6 which is a GUI based on QoS supported the client side QoS for Web Service applications is the ability of their services to provide added value to the best solution for requesters’ enquiries, with a specific requirements. QoS parameters help determining which of the available Web Services is the best and meets clients’ requirements. In a previous work we used two datasets one of them consists of 365 web services, and the other one contains more than 2000 web services ,two databases were built using data of these two datasets on MySQL Server.

Description of QWS dataset : QWS dataset consists of different rows of web service implementations and their attributes as presented below (http://www.uogue/ph.ca/~qmahmoud/qws/index.html). The attributes used in our dataset are as follows:

1. Response Time: time taken to send a request and receive a response (ms) 
2. Availability: number of successful invocations/total invocations (%) 
3. Throughput: total number of invocations for a given period of time (#/sec) 
4. Successability: number of response / number of request messages (%) 
5. Reliability: ratio of the number of error messages to total messages (%) 
6. Compliance: extent a WSDL document follows WSDL spec. (%) 
7. Best Practices: extent a Web service follows WS-I Basic Profile (%) 
8. Latency: time taken for the server to process a given request (ms) 
9. Documentation: measure of documentation (i.e. description tags) in WSDL (%) 
10. WSRF: Web Service Relevancy Function: a rank for Web Service Quality (%) 
11. Class: levels representing service offering qualities(1 through 4). 
12. Name: service name. 
13. WSDL: WSDL file location.

Fig(7) is a flow chart represents the discovery process. Client could select between all Web Services due to a specified URL.
or a specified function then he/she could query from the returned result using QoS parameters to get Web Service that fulfills his/her requirements. The dashed area represents the discovery process, client could select between all Web Services due to a specified URL or a specified function then he could query from the returned result using QoS parameters to get the best result.

Fig 7: A flow chart for Web Services’ Discovery

4. EXPERIMENTAL WORK

The experimental work was done in two steps (i) Implementing Web Service’s Classifier System, testing the results (ii) Building a Graphical User Interface (GUI).

4.1 WS-Classifier System

<table>
<thead>
<tr>
<th>Table 1. Software Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local Server</strong></td>
</tr>
<tr>
<td><strong>Programming Language</strong></td>
</tr>
<tr>
<td><strong>DataBase Management System</strong></td>
</tr>
<tr>
<td><strong>Programming Environment</strong></td>
</tr>
<tr>
<td><strong>GUI design</strong></td>
</tr>
</tbody>
</table>

Connecting between WAMP Server and Netbeans environment through MySQL driver as a local host and classifying data imported from a dataset contains 365 web services using SQL statements and phpmyAdmin. The output from the classifier system was 221 categories when classifying data due to URL and similar functions fig.(8).
Using the other dataset containing more than 2000 Web Services, the output from the classifier system was 28 categories when classifying data due to URL and similar functions, it becomes more specified and easily dealing with as it reduced from 221 category to 28 only.

Table (2) displays Web Services’ Capacity due to URL, first column represents the output from the experimental results using two different datasets (dataset1 which contains 365 different web services, and the second column using dataset2 contains 2500 different Web Services), the third one shows the percentage of the differential between them.

<table>
<thead>
<tr>
<th>URL</th>
<th>No of Web Services (Dataset1)</th>
<th>No of Web Services (Dataset2)</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.com</td>
<td>233</td>
<td>745</td>
<td>3.19</td>
</tr>
<tr>
<td>.de</td>
<td>64</td>
<td>613</td>
<td>9.58</td>
</tr>
<tr>
<td>.net</td>
<td>57</td>
<td>151</td>
<td>2.65</td>
</tr>
<tr>
<td>.org</td>
<td>16</td>
<td>167</td>
<td>10.44</td>
</tr>
<tr>
<td>.gov</td>
<td>7</td>
<td>40</td>
<td>5.7</td>
</tr>
<tr>
<td>.edu</td>
<td>7</td>
<td>57</td>
<td>8.14</td>
</tr>
<tr>
<td>.info</td>
<td>6</td>
<td>21</td>
<td>3.5</td>
</tr>
<tr>
<td>.dk</td>
<td>4</td>
<td>168</td>
<td>42</td>
</tr>
</tbody>
</table>
4.2 A Graphical Web Based Interface (GUI)

Query forms presented on HTML pages are the only interfaces that users can access the content hidden in online databases, and therefore they are also called deep web or hidden web. As we assumed before that the future image of Web Services' Discovery will be through the web depending on the model of authors at[6], we constructed a web site that will be an a login page fig(13):

![Web Services' capacity Comparison due to URL](image1)

**Fig 10:** Changing in Web Services' Capacity due to URL

**Table 3. No of Web Services capacity due to their function**

<table>
<thead>
<tr>
<th>Function</th>
<th>Web Services' Capacity (Dataset1)</th>
<th>Web Services' Capacity (Dataset2)</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculator</td>
<td>3</td>
<td>13</td>
<td>4.33%</td>
</tr>
<tr>
<td>Code</td>
<td>12</td>
<td>57</td>
<td>4.75%</td>
</tr>
<tr>
<td>Email</td>
<td>4</td>
<td>13</td>
<td>3.25%</td>
</tr>
<tr>
<td>Fax</td>
<td>3</td>
<td>3</td>
<td>0%</td>
</tr>
<tr>
<td>Phone</td>
<td>7</td>
<td>16</td>
<td>2.29%</td>
</tr>
<tr>
<td>Search</td>
<td>11</td>
<td>56</td>
<td>5.1%</td>
</tr>
<tr>
<td>SMS</td>
<td>6</td>
<td>23</td>
<td>3.83%</td>
</tr>
<tr>
<td>SOAP</td>
<td>11</td>
<td>1018</td>
<td>92.6%</td>
</tr>
<tr>
<td>Weather</td>
<td>6</td>
<td>14</td>
<td>2.33%</td>
</tr>
<tr>
<td>Airlines</td>
<td>2</td>
<td>2</td>
<td>0%</td>
</tr>
</tbody>
</table>

![Web Services' Capacity due to Similarity in function](image2)

**Fig 11:** Web Services capacity due to similarity in function
After logging in the client will be able to access the GUI fig.(14) which will enable him/her to display Web Services due to URL or due to the function of the Web Service as follow:

**Fig 13:** Login Page

**Fig 14:** The output from Gov URL Search

For example when client asked for “search” Web Service, he/she can easily select search from the list and the result will be like this:

**Fig 15:** The output from search Web Service
Which shows that there are eleven Web Services for “search” and now it is the role of QoS to select between the different Web Services that have the same functions. Advanced search criteria is applied to the web site, client can easily enter his/her QoS requirements easily as shown fig.(16):

![Fig 16: The advanced search for QoS](image1)

Figure(17) shows the output from the advanced search which client can use after having different services with a similar function and he/she has to select between them due to QoS requirements. Using interface in fig.(17) Web Service consumer could enter restricted values of QoS parameters to get the Web Service that fulfills his/her requirements.

![Fig 17: The advanced search after selecting QoS values](image2)
Fig 18: The output from the advanced search

Print the page criteria is also applied to the web site as seen in fig.(19)

Fig 19: Print the page criteria

Data could be exported fig(20) to other programs like word, excel and other

Fig 20: the Export criteria

5. CONCLUSION

From UDDI to Search Engine in the Web Services’ Discovery process and merging between them is the future image. Classifying Web Services due to URLs they are published in and grouping them due to similarity in their functions before storing them in WSS could enhance Web Services discovery process. Constructing a web based interface (a client GUI) that could be used by the web services consumer to select between Web Services with the same function with respect to their QoS parameters will facilitate the Web Services’ Discovery.
6. FUTURE WORK

After Classifying process a new structure of database depending on the relational entities and building an infrastructure for controlling the storage and retrieval processes will be implemented. Searching for Web Services based on QoS parameters, schema properties, service reputation, trust, and semantic matching will considerably increase the relevancy of finding and selecting appropriate Web services. Some essential QoS parameters were missed from the dataset like cost, which must be added to it. Studying the ability of frequently updating QoS values of web services must be studied in a future work. At the other side for consumer constructing Open Web Directories contain all published Web Services and their quality of services (QoS) will enhance the WS discovery process, and facilitate it for the consumer to easily fulfill his requirements.

7. ACKNOWLEDGEMENTS

I take this opportunity to express my deep gratitude and regards to Professor Dr. Hesham Arafat and Dr. Ahmed Saleh my thesis supervisors, for their incomparable guidance, supervision and constant encouragement throughout the course of this research. I would also like to thank Dr. Eyhab Al_Masry for his advices and providing me with datasets for the experimental work.

8. REFERENCES

[8] Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI) by Qusay H. Mahmoud, April 2005.
[9] Efficient Web Service Discovery and Composition using Constraint Logic Programming Srividya Kona, Ajay Bansal, Gopal Gupta and Thomas D. Hite 2 Department of Computer Science The University of Texas at Dallas Metallect Corp.
[22] WSCE: A Crawler Engine for Large-Scale Discovery of Web Services Eyhab Al-Masri and Qusay H. Mahmoud.
[29] WSCE: A Crawler Engine for Large-Scale Discovery of Web Services Eyhab Al-Masri and Qusay H. Mahmoud Department of Computing and Information Science University of Guelph, Guelph, Ontario, Canadaemailmasri,qmahmoud)@uoguelph.ca
[31] Semantics in Service Discovery and QoS measurement Chen Zhou, Liang-Tien Chia, and Bu-Sung Lee.
[33] Using Common Sense Reasoning to Enable the Semantic Web Alexander Faaborg , Sakda Chaiwarorawitkul , Henry Lieberman, MIT Media Lab.