VOL. 9, NO. 4, APRIL 2014 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

A MULTI-AGENT-BASED QOS-DRIVEN WEB SERVICE DISCOVERY AND COMPOSITION FRAMEWORK

Manoranjan Parhi[1], Binod Kumar Pattanayak[1] and Manas Ranjan Patra[2]
[1]Depatrment of Computer Science and Engineering, ITER, Siksha 'O' Anusandhan University, India
[2]Department of Computer Science, Berhampur University, India
E-Mail: mrparhi@gmail.com

ABSTRACT

Web service has been playing a magnificent role in the field of application development. It facilitates giving a global touch to standalone applications components to interact with each other via interfaces and form larger application systems. Web service users who seek for single web service or composition of web services face a lot of problems regarding its discovery. Discovery of suitable web services has become a challenging issue due to the increasing number of selection parameters and constraints. This paper deals with a hybrid multi-agent based web service discovery mechanism which involves an artificial intelligence approach to efficiently interpret the user requirements based on both functional and non-functional demands and fuzzy constraints. The proposed model utilizes the services of intelligent software agents. Some of the agents like the reputation agent analyses the popularity of web services and assign ranks to the web services based on user feedback and statistical information. The behavior of individual user is being tracked, from which the intelligent agent interprets the fuzzy requirements of users through set of logical and analytical calculations whereas the composition agent provides flexibility to the user for custom composition of web service packages using different individual web services. In brief, we present an automated customer-centric web service discovery and composition approach which aims in an efficient web service discovery and composition followed by customer satisfaction using multi agents.

Keywords: web services, quality of service, multi-agent systems, fuzzy logic, service composition.

1. INTRODUCTION

World Wide Web has become the most popular weapon in today's era of innovations and technologies. In [1], the authors stated that, WWW is a collection of human readable pages that are virtually unintelligible to computer programs and it has only been restricted as a global repository of digitized information, this information is, by and large, unavailable for autonomous computation where in the computations are carried out without any human intervention. A web service is a public interface of an application which can be invoked remotely to perform a business function or a set of functions. Web service has been defined as self-contained, self-describing, modular application that can be published, located and invoked across the web. According to [2], a web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.

Web Services allow applications to be integrated more rapidly, easily and less expensively than ever before. Integration occurs at a higher level in the protocol stack, based on messages centered more on service semantics and less on network protocol semantics, thus enabling loose integration of business functions. These characteristics are ideal for connecting business functions across the web both between enterprises and within enterprises. They provide a unifying programming model so that application integration inside and outside the enterprise can be done with a common approach, leveraging a common infrastructure. The integration and application of Web Services can be done in an incremental manner, using existing languages and platforms and by adopting existing legacy applications. Moreover, Web Services compliment Java 2 Platform, Enterprise Edition (J2EE), Common Object Request Broker Architecture (CORBA) and other standards for integration with more tightly coupled distributed and non distributed applications. Web Services are a technology for deploying and providing access to business functions over the Web; J2EE, CORBA and other standards are technologies for implementing Web Services.

With an increasing number of Web services providing similar functionalities, more emphasis is being given on how to find the service that best fits the consumer's requirements. In order to find optimized services, the service consumers and/or discovery agents need to know both the functional and non functional attributes of web services. The problem, however is that the current UDDI registries do not provide flexibility for service providers to publish the QoS information of their services, and the advertised QoS information of Web services is not always trustworthy.

The main objective of this paper is to provide a customer-centric multi-agent based hybrid web service discovery approach which involves dynamic web service selection followed by providing flexibility towards web service composition on the basis of both functional as well as non-functional aspects of web service.

In our work we have used Quality of Service parameters which allows user to discover relevant web services as per his/her preference in order to gain optimal results followed by achieving boost in time by reducing

the search space due to implementation of distributed service oriented architecture.

We have considered four types of QoS parameters in our work which are: Cost, Reliability, Response Time and Availability.

- **Cost:** Cost involved in the request and usage of a Web service.
- **Response time:** Time taken to send a request and receive a response.
- **Availability:** is the probability that system is up and can respond to consumer requests.
- **Reliability:** Ratio of the number of error messages to total messages.

1.1. Web service architecture

Web service architecture defines interactions between three roles: service provider, service registry and service requester.

- A "Service Provider" is a network node that provides an interface to the web services it is providing; it also responds to requests for using its services.
- A "Service Requester" is a network node that discovers and invokes web services to realize a business solution.
- A "Service Registry" or service broker is a network node that acts as a repository, for the description of services interfaces that are published by service providers

The interactions between these roles involve three operations: publish, find and bind.

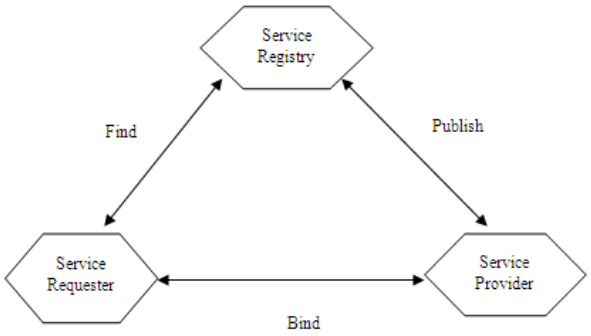

Figure-1. Web service architecture.

1.2. Fuzzy constraints

A fuzzy concept is a varying concept which does not have a specific definition rather it may considerably vary as per underlying conditions. In other words, one can define fuzzy logic as a vague concept which depends on individual view of perception. Those concepts lack clarity and are very difficult to operationalize and debug.

Due to the high proliferation of web services, selecting the best services from functional equivalent service providers have become a real challenge, where the quality of the services plays a crucial role. But quality is uncertain, therefore, several researchers in [3, 4, 5, 6] have applied Fuzzy logic to address the imprecision of the quality of service (QoS) constraints. Furthermore, the service market is highly dynamic and competitive, where web services are constantly entering and exiting this market, and they are continually improving themselves due to the competition. Current fuzzy-based techniques are expert and/or consensus based, and therefore too fragile, expensive, non-scalable and non self-adaptive.

1.3. Importance of multi-agent systems

Multi Agent System is a promising sub field of artificial intelligence which is mainly concerned with interaction of intelligent agents to solve a common issue. As per [7], an intelligent agent can be defined as a computer system situated in some environment, and able to perform autonomous actions in order to achieve its design objectives.

The multi agent system is a loosely coupled network of problem-solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity. Multi agent system relied on semantically rich semantic representation and techniques for communication among agent communities. The importance of agent technology has been realized as one of the important technologies to successfully support the activities like e-business which require autonomous entities to handle dynamism of communication and results in conceptual simplicity and enhance scalability.

Agent systems are self-contained software programs embodying domain knowledge and having ability to behave with a specific degree of independence to carry out actions needed to achieve specified goals. They are designed to operate in a dynamically changing environment. Agents typically include a set of features. The main features of agents include autonomy, pro-activity, re-activity, communication and cooperation, negotiation, learning etc.

According to [8], a software agent, which is sometimes known as autonomous and intelligent agent, is defined as a computer program which takes a special function in representation of other resources in systematic computing environments.
The basic attributes of a software agent are:

- They are not strictly invoked for a task, but activate themselves.
- They may reside in wait status on a host, perceiving context.
- They may get to run status on a host upon starting conditions.
- They do not require interaction of user.
- They may invoke other tasks including communication.

Multi Agent System symbolizes a novel way of analyzing, designing, and implementing complex software system. In Multi Agent System communication is the

fundamental element for interaction and it facilitates the social organizations in reorganizing their agents to cooperate and coordinate their actions. A Multi Agent System is a system composed of multiple interacting intelligent agents within an environment. Multi Agent System can be used to solve problems that are difficult or impossible for an individual agent to solve.

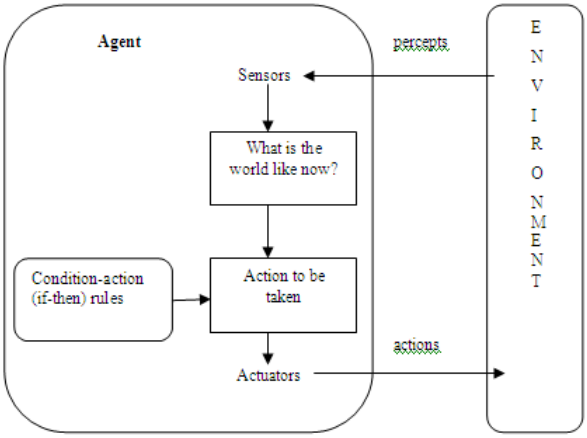

Figure-2. Architecture of an intelligent agent.

A MAS has the following advantages over a single agent.

- A MAS distributes computational resources and capabilities across a network of interconnected agents so it does not suffer from single point failure and bottleneck.
- A MAS allows for the interconnection and interoperation of multiple existing legacy systems.
- By building an agent wrapper around such systems, they can be incorporated into an agent society.
- An MAS models problems in terms of autonomous interacting component-agents, which is proving to be a more natural way of representing task allocation, team planning, user preferences, open environments, and so on.
- An MAS efficiently retrieves, filters, and globally coordinates information from sources that are spatially distributed.
- An MAS enhances overall system performance, specifically along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility and reuse.

The structure of this paper is as follows: The section 2 briefly introduces major related works in the area of Web service discovery and composition based on QoS. Section 3 explains the structure of the proposed framework. Accordingly, section 4 presents the service registry and discovery algorithms. After that, the implementation is represented in section 5, followed by discussing the advantages of the proposed model in section 6. Finally, we conclude the whole paper and provide dimension towards future work in section 7.

2. LITERATURE REVIEW

Several researches have been done on web service discovery and composition by using various approaches. According to [9], there are two challenges facing the practicality of Web services discovery:

(a) Efficient location of the Web service registries that contain the requested Web services.

(b) Efficient retrieval of the requested services from these registries with high quality of service (QoS).

In our review, we have considered the survey of those papers which are related to web service discovery and composition basically based on QoS.

2.1. Related works on web service discovery

Three layers matching strategy has been discussed in [10] for semantic web service discovery based on user preference and QoS. They have provided flexibility to the users to set the weights in order to realize their personalized choices.

Similarly in [11] a framework is presented for execution of semantic web services using context-aware broker agent. In their framework, context information are used by software agents to identify web services based on situations in order to improve effectiveness and gain efficiency in providing services to clients.

In [2] the researchers have discussed an agent based approach for web service selection by using the wsdl information provided by the service provider using efficient data mining tool weka. They have included considering user's feedback and QoS rating methodology for efficient web service discovery but as the QoS values are not specific and involves too much approximation, sophisticated users may face difficulty towards knowing the accurate QoS values associated with any web service.
To invoke and access semantically enriched web services in the field of mobile E-commerce, a technology has been proposed in [1], which facilitates wireless users to use semantic web services even in the absence on online service requestor by the use of automated mobile software agents.

An extended service discovery model has been proposed in [12] involving traditional components like service provider, service consumer and UDDI along with a new component known as certifier which verifies the QoS of the services prior to registration. . The consumer can also verify the advertised QoS with the Certifier before using to a Web service. Although this model incorporates QoS into the UDDI, it does not integrate consumer feedback into the service discovery process.

Later, an agent based framework has been discussed in [13] for autonomic web services. They have embedded an autonomic manager for controlling the state of web services and used five software agents namely a planning agent, an execution agent, a composition agent, a discovery agent, and a monitoring agent for incorporating their multi-agent system. However, the implementation of autonomous manager has not been discussed, so the

feasibility of the work could not be judged completely. A similar kind of multi-agent system has been proposed in [14] which is capable of efficient web service discovery along with QoS registration, verification, certification and confirmation by implementing several multi-agents like response agent, certification agent and query agent in their work.

Distributed web service discovery architecture is presented in [15], which is designed to be scalable, flexible and reliable. It is based on the concept of centralized shared space and intelligent search among a subset of spaces. Since it does not discusses about automation in search process it may not result into an efficient search process.

The authors in [16] have presented an integrated approach of Artificial Intelligence specially data mining technologies and service rating technologies into SOA architecture for service discovery. A semantic web service discovery approach has been discussed in [17] using SPARQL for expressing the functional aspects of web services like pre-conditions and post-conditions. Though, the presented technology is inspiring but the authors have not discussed any of the non-functional attributes of web services like performance and QoS.

The researchers in [18] have proposed a framework for reputation oriented semantic web service discovery where ratings of service on various contexts are collected by service consumer using a reputation management system. A framework is proposed in [19] that uses user preferences as a searching tool for selection and the system ranks the available services on the basis of requirement.

A two phase semantic based web service discovery technique has been discussed in [20], which is capable of discovering services in an efficient way by implementing a two-level matchmaking technique, namely operational matchmaking and operation-composition matchmaking. Another semantic web service discovery model has been presented in [21] where the searching approach discussed by the authors involves an Artificial intelligence technique where a double level filtration is used by the requester agents for service discovery. However, the economic efficiency, performance with respect to time and scalability of the model has not been discussed.

2.2. Related works on web service composition

As per web service composition is concerned, it is a very challenging field. Hence, a lot of approaches have been discussed by many researchers by following various methodologies some of which are reviewed by us in our literature work.

A web service composition technique has been discussed in [22] on the basis of multi-agent negotiation using Contract-Net protocol. Each elementary Web service has an agent responsible to it. The objective of these agents is to find out the best Composite QoS. For the composition process, they have used Case-Based Reasoning (CBR) technique. However, the above discussed model is based on local optimization that means it does not guarantee best QoS for a composite web service set.

A runtime service discovery and composition approach has been discussed in [23] which uses backward context based service selection algorithm. The main idea behind the algorithm is that, service discovery is occurred step by step. After discovering services at each step, the algorithm goes back and checks if the selected services from previous step are best for composition or not and then it invokes them.

In [24], authors have described a composition strategy by analyzing multiple execution paths of a composite service which are specified using UML (Unified Modeling Language) state chart. They have used linear programming by integrating the local optimization approach and global planning approach for composition problem.

Similarly a AgFlow platform has been presented in [25], where web service composition is done by computing the optimized QoS values by incorporating user requirements. In [26], the authors have proposed a web service composition approach by using integer programming and constraint programming approaches for optimizing QoS by leveraging constraints hierarchies as a formalism to represent user constraints.

Apart from the classical composition approaches, various nature inspired algorithms are also used for composition purposes. As per the work stated in [27], the authors have proposed an algorithm named as ACAGA_WSC. This algorithm is a combination of ant colony and genetic algorithm which solves service composition problem with a greater efficiency.

Using genetic algorithm, another approach has been undertaken in [28]. The authors have used fitness function where constraints are used and all form of workflow in business processes are taken into account. The approach used by the authors for calculating overall QoS is quite similar to the composition technique used in [29].

Some of the researchers have used discrete particle swarm optimization algorithm for web service composition. In this method, each particle is considered as a solution having its own position and velocity. Such concept has been discussed in [30], where particles try to change their positions according to two elements: first their last best position and second the best position that has been seen so far. Though discrete particle swarm algorithm proves to be an efficient approach in terms of service composition in terms of optimality but it is not scalable in terms of time as the execution time is very slow.

3. PROPOSED FRAMEWORK

The various aspects of web service discovery approaches inspired us to design a new framework based on syntactic web service discovery technique. The proposed framework is based on distributed architecture. According to [31], although centralized registries can

provide effective methods for the discovery of Web services, they suffer from problems associated with having centralized systems such as a single point of failure, and bottlenecks Distributed architecture provides wide implementation by increasing the resources utilization. The performance of distributed architecture is much more efficient than centralized architecture. Customers can search for the desired web service more efficiently as distributed architecture provides concurrent query handling.

Use of agents divides the work load of the web service discovery into different independent tasks. As our framework is based on distributed architecture, the use of different intelligent software agents benefits the performance of discovery process by effectively dividing the task into many number of independent sub tasks which are efficiently handled by those software agents.

We have focused on the Quality of Service parameter to effectively search for a desired web service. The QoS requirements for Web services are more important for both service providers and consumers since the number of Web services providing similar functionalities is increasing. In [32], the authors stated that, if the discovery engine returned multiple candidates Web services which provide the same functionality, then Quality of Service (QoS) is becoming an important criterion for selection of the best available Web service.
Apart from including the QoS parameters, we have also considered the fuzzy constraints. From our research work, we have found in many papers that use of only QoS parameter is not an effective solution to the problem statement. For example, a customer who wants a Weather Forecast web service searches for the required service with the following Quality of Service requirements:

a) Availability- 89%
b) Response Time- 3ms
c) Reliability- 80%
d) Cost- 450$

As per the input criteria, if no service is found then the customer doesn't get any result. If in case, the registry contains any web service say X, having the QoS parameters as follows:

a) Availability- 92%
b) Response Time- 3.1ms
c) Reliability- 80.5%
d) Cost- 448$

But as a fuzzy constraint has not been taken into account, Web Service X is not returned as a suggestion to the customer. Use of fuzzy constraints increases the efficiency of the web service discovery approach by providing the customers the web services which are not actually satisfying the input QoS constraints, but are close to the QoS constraints specification.

In the proposed framework, each registered web service is associated with a score value, and user rating specifications. To provide the service consumer, the popularity information of different services and to help the customer to select the best service among many number of similar web services, we have used a reputation agent, which compares each web service and keep track on their popularity from which the score value of each web service is computed. This score information will highly help the consumer during web service selection. Being a customer centric framework, it provides service provider and consumer to rate and provide feedback any web service, which information will be recorded and used to manipulate the score attribute associated with each service and its popularity.

The framework also provides the facility to compose web services as per the consumer requirements. Many cases arise when any customer needs more than one web service in a package. In such a case if such webs service package is not found as a standalone web service, then the model provides the consumer the flexibility to compose a web service package as per his/her choice.

For example: Suppose a customer needs a Travel package web service which involves basically three features:

a. Transportation service
b. Hotel service
c. Tourist guide service

On searching for Travel package web service if no result is returned, then the customer can search for optimal solutions of Transportation, Hotel and Tourist Guide web service separately and then he/she can compose a new web service package using those individual services. The score and QoS parameter value for the composed web service will be calculated accordingly and if the web service package proves to be popular and useful then it will be registered as a composed web service in the registry which can be used further.

There are two external entities that are associated with the proposed framework. These are service consumer and service provider. The complete framework aims in basically three functionalities:

a. Effective web service discovery
b. Effective web service registration
c. Effective service composition

The discussed model uses various agents to handle user/ provider request concurrently and provide efficient results in short span of time. There are various intelligent agents acting in the discussed framework which work concurrently as individual agents having no or very less dependency among each other. Those agents exchange information and data among each other and work accordingly.

The various agents used in the discussed framework are as follows:

3.1. Discovery agent

This agent involves Discovery of the requested web services from the Service Registry using the information provided by service consumer. We believe that it is not possible that all the service providers have entered the Quality of service information while registering the services. There might be also possibility that a user may not be requiring to feed the Quality of Service information while searching for a web service. For such cases, the discovery agent provides two ways for searching mechanism.
Those two approaches are as follows

a) Discovery without QoS parameters

Here, the service consumer has to feed the search keywords in the search field. There is no need to enter any Quality of Service parameters for the searching. Here the searching will be done on the basis of keywords only based on only functional specifications.

b) Discovery with QoS parameters

Here, the service provider has the flexibility to enter the keywords in the search field along with Quality of Service parameter and fuzzy constraints. The searching in this case will be more specific and accurate. Here, the discovery agent searches for the service using both functional and non-functional information.

3.2. Reputation agent

This agent plays a very vital role in this framework. It keeps tr ack of the reputation and popularity of individual web services from user feedback and responses. Each web service has its own functional and non functional specifications. Based on the information provided by the service provider during web service registry, the initial score of the web service is calculated. As the web service is consumed by the consumer, and due to the availability of new similar kind of web services, the score value is modified. Moreover, whenever a user provides feedback and rate a particular web service, then its score value is altered. The score value is dynamically calculated as follows-

Suppose a set of service $S= \{S_1, S_2,\ldots,Sn\}$, there are m QoS attribute $\{q_1, q_2,\ldots,q_m\}$, its weight is $\{w_1, w_2,\ldots,wm\}$. Users can choose the QoS attributes they need, and can set the weight of QoS attributes based on their preference. This can reflect the user's personalization options. If user think the attribute qi is important, so he can set the weight of qi higher to reflect user's demand.

For the quality of service, we usually use the positive indicator and the negative indicator to measure the quality of service. Positive indicators (positive) means that the value the greater the quality of service the better, such as reliability; negative indicators (negative), the value is the higher the quality of service the worse, such as cost, response time. We can use 5 quantify QoS attribute. Then, we can get the qualified QoS attribute $\{q'_1, q'_2,\ldots, q'_m\}$ as follows:

$$q_i = \begin{cases} q_i / Max(q_1 \ldots q_m) & \rightarrow Positive \\ Min(q_1 \ldots q_m) / q_i & \rightarrow Negative \end{cases}$$

3.3. Composition agent

This agent facilitates service consumer to compose a web service package as per their need and request the custom package to be added to the service registry as a composite web service package for future use.

Many cases arise when any customer needs more than one web service in a package. In such a case if such webs service package is not found as a standalone web service, then the proposed framework provides the consumer the flexibility to compose a web service package as per his/her choice.

For example: Suppose a customer needs a Travel package web service which involves basically three features-

a) Transportation service
b) Hotel service
c) Tourist guide service

On searching for Travel package web service if no result is returned, then the customer can search for Transportation, Hotel and Tourist Guide web service separately and then he/she can compose a new web service package using those individual services. The score and QoS parameter value for the composed web service will be calculated accordingly and if the web service package proves to be popular and useful then it will be registered as a composed web service in the registry which can be used further.

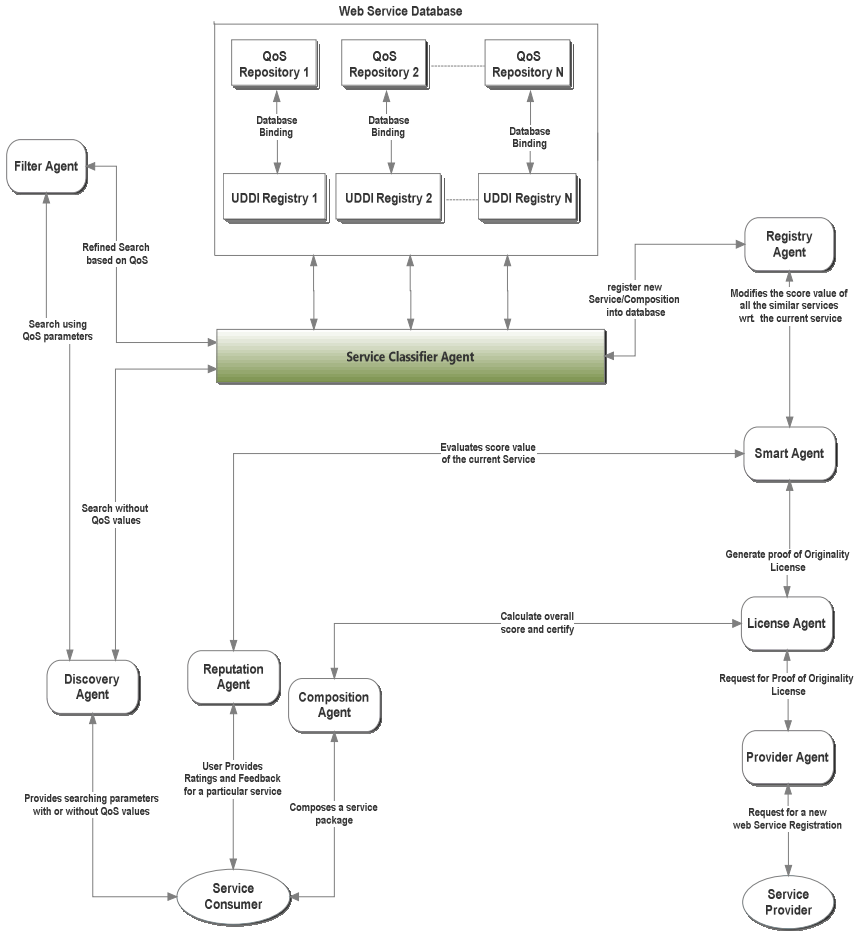

Figure-3. Proposed framework.

3.4. Filter agent

Filter Agent performs a refined web service searching technique using score value and QoS parameter preferences provided by the service consumer. Whenever service consumer searches for a web service by providing the required Quality of Services parameter and fuzzy constraint value, then the discovery agent first searches the registry on the basis of syntactic search based on only keywords. After the list of services retrieved from the registry, the filter agent refines the list and removes those services that violate the quality of service specification and fuzzy consideration value provided by the consumer. Only those services that satisfy the query and constraints provided by the service consumer are returned back to the consumer.

3.5. Provider agent

This agent helps the service provider to register a new service or to upgrade an existing service. Provider agent has a direct communication with the service provider. A service provider can register a new service and upgrade an existing service by the assistance of provider agent. A service provider basically has the following operations-

a) To register a new service
b) To upgrade an existing service
c) To keep track on a service popularity from user feedback
d) To update the service feature and functionality from time to time

3.6. License agent

As we know that the information provided by the service providers regarding a particular web service may vary from the actual values, so this agent helps in certifying the web services information as original and genuine by performing various QoS analysis process so that the consumer may not be deceived. Many a time, for the sake of advertisement the service providers provide wrong feature and faulty information in the service description to gain popularity due to which the service consumer suffer. Thus, the license agent holds the responsibility to check and verify the provided information by using any performance or Quality of service analysis tool, so that it can be tested if the information provided by the service provider is correct.

3.7. Smart agent

This agent is responsible to update the score value of each related web services whenever the following cases occurs-

a. A new web service is registered
b. An existing web service is modified.
c. A new composition is made.
d. Whenever a web service looses popularity due to availability of new technologies or it is outdated.
e. A user rates a web service or provides feedback.

Score value calculation needs to be dynamic. The calculation of score value on the basis of only Quality of Service information is not worthy as we know that besides the functional features, a web service is associated with a lot of non-functional features like popularity, brand/ service provider reputation, customer feedback, customer ratings and critics, time etc. Thus, the smart agent in the proposed framework plays a very vital role in this context. No doubt, whenever a new web service is registered then on the basis of the QoS information provided by the service provider, its score is calculated. But after a web service has been registered and customer provides his/her critics and ratings then also the score value needs to be modified. Whenever a new service is registered which proves to be superior then the score of the other related web service needs to be degraded. Such type of dynamic score calculation is being done by the smart agent which proves to be very beneficial in this web service discovery approach on both service provider and service consumer point of view.

3.8. Registry agent

This agent holds the responsibility to register a web service which is certified as genuine into the web service registry by classifying them into a specific category. As we have considered a distributed framework, the web services are classified into different categories so that different categories of web services can be distributed among different UDDI which are scattered all over the distributed network. The registry agent works basically for registering a new web service in the service registry. Whenever the is any modification in a web service from any side i:e service provider or service consumer side, then it is the work of the registry agent to store such information in the registry. Each time a new service is registered or an existing service is modified then the registry agent stores the updated information in the service registry.

3.9. Service classifier

The service classifier is used to utilize the advantages of distributed architecture in an efficient way. The main objective of service classifier is to classify the web services into different categories. As we know that there will be a lot of web services in the service registry and it will be a troublesome and inefficient way to search the whole service registry for a particular web service. So, the service classifier is directly linked with a classifier registry where all web services are categorized. So, whenever a query for a particular web service is received then first of all the service classifier determines the category to which the service belongs, then it searches for the web service in only those UDDI where web services of that particular category can be found. This process reduces the search space and also increases the efficiency and performance of web service discovery process. There must be high security to protect the service classifier module from any unauthorised attack and failures as it is the base of web service discovery process.

3.10. Web service database

The proposed framework is based on distributed architecture. Distributed architecture provides wide implementation by increasing the resources utilization. The performance of distributed architecture is much more efficient than centralized architecture. Customers can search for the desired web service more efficiently as distributed architecture provides concurrent query handling. The web service data are scattered in a distributed network so that more storage space can be achieved and there should be no single point failure and bottleneck.

Unlike the classical web service discovery technique, here in this framework, there is no centralized UDDI registry. Here, there are various UDDI registries that are scattered in different geographical areas in a distributed network. In each node or system in the distributed network, there is another registry known as Semantic Repository containing semantic information of the web services like keywords associated with the web service, the score values, the feedbacks, ratings, popularity score and the quality of service information. Both the UDDI registry and the Semantic repository are bided to each other and exchange of data and information are done between them.

4. ALGORITHMS FOR SERVICE REGISTRY AND DISCOVERY

4.1. Service registry algorithm

Steps

a) Click the member login button in the search page.
b) If the service provider is not registered then GOTO step 3 else GOTO step-6.
c) Insert the username and password and click Login.
d) Click New Provider.
e) Insert the Service Provider name, user ID, password and click register.
f) Insert the new service details like service name, service category, availability, reliability, response time, cost and feature.
g) Read the website declaration
h) Check the declaration check box.
i) Press the Button “Request for verification”.
j) End

4.2. Service discovery algorithm

Steps

a) Insert the web service name or the web service related keyword into the search box.
b) Hit the search button
c) If search box text is matched with any web service record then GOTO step-4 else step-5.
d) Fetch the matched web service records from the database and show to the service consumer then go to step-6.
e) Print no results found then GOTO step-9.
f) Show the Quality of Service entry box and the fuzzy constraint box to the consumer.
g) Enter the QoS constraints including fuzzy specification for refined search and click filter search button.
h) Fetch the filtered web service records from the database and show to the service consumer.
i) End

5. PRACTICAL IMPLEMENTATION

We have implemented our framework in a web based application project named as "Wizardisc". This project is a working model to show the feasibility of this model and it elaborates all the functionalities of the proposed framework in a well mannered and specific way. We have used Microsoft asp. Net to design this project and Microsoft SQL Server 2008 to implement the databases used in this project as our back end.

Figure-4. Wizardisc homepage.

The following GUI prompt stated in Figure-4 is the Wizardisc Homepage where both Service Provider and Service Consumer can search and register web services. The search box provided on top of the web page is for web service searching purpose.

On opening the homepage the repeater control used in this webpage shows all the popular web services registered. Each row in the repeater control is designed in such a way that any service consumer can have a clear idea about a particular web service by just viewing it.

Each row in the repeater control shows the following information:

a) Web service name
b) Web service provider
c) QoS information like cost, availability, response time and reliability
d) User ratings and reviews
e) Features
f) Score value

Once a service consumer search for a web service for the first time, then the searching is done only on keywords basis. After that the QoS panel is provided to the service consumer to refine his/ her search by entering the required Quality of Service parameters and fuzzy constraints value.

We have considered two approaches for the searching process which are as follows:

Figure-5. Searching without QoS.

a) Searching without QoS for novice Service Consumer as stated in Figure-5.
b) Searching with QoS for sophisticated Service Consumer as stated in Figure-6.

For selection of QoS parameters, we have used Ajax Range Selector in Figure-6, where the service consumer can feed a range of QoS value to the system which will be very helpful for the service consumer to select a best service from a wide range of suggested services.

The fuzzy constraint box in Figure-6 is optional which can be used by the service consumer who is not getting a web service as per the provided information. Fuzzy parameters make aware to the consumer about those services which do not exactly match the requirements of the consumer but are close to the provided requirements.

Figure-6. Searching with QoS and fuzzy logic.

Figure-7 shows the details information of a web service. Any web service when clicked in the homepage, it will redirect to a detailed view page where all information regarding a web service is displayed.

Figure-8 shows the feedback page where any user can view the feedbacks and critics associated with any web page and can also post his/ her feedback. Provision for user rating is provided so that user can rate a particular web service to affect its demand and popularity.

The service registration page has been shown in Figure-9 where service provider can register a new service by providing the QoS attributes which will undergo background verification as shown in Figure-10. Here the service provider has to agree the terms and conditions of the "Wizardisc" confirming that the website will verify the information provided by the service provider prior to add the new service in the web service registry and make it available to customers.

Figure-7. Detailed views.

Figure-8. Feedback page.

Figure-9. Service registration page.

Figure-10. Verification page.

6. ADVANTAGES OF PROPOSED MODEL

a) Distributed architecture provides concurrent query handling.
b) Use of agents facilitates better load sharing and load balancing.
c) Use of QoS parameters highly benefits the searching mechanism.
d) Use of fuzzy parameters enhances the searching technique.
e) Score calculation is dynamic.

f) User feedback and ratings are taken into account in score calculation.
g) Provider can upgrade any service and can stay up to date.
h) Service Provider can keep track on the service popularity.
i) User's requirements can be easily known from their feedback.
j) Composition technique is very easy to understand providing a larger searching domain to the service consumer.
k) Classification of web services reduces the search space and increases the performance of searching process.
l) Certifier agent and smart agent keep the services score value always updated and the score calculations are done on regular basis.

7. FUTURE WORK AND CONCLUSIONS

The proposed framework is an automated customer-centric web service discovery and composition approach which aims in an efficient web service discovery and composition followed by customer satisfaction using multi agents. As discussed above, the proposed framework tries to provide adequate advantages over classical web service discovery approaches by implementing a lot of new technologies and methodologies to achieve optimal customer satisfaction. We have found out certain limitations in our framework which we have planned to eradicate in our future work. In our future work, we will be more concerned about providing security of data and will aim in designing a well defined failure handling mechanism for the same. The basic objective of our future work will be based on making full utilization of the advantages provided by Multi- Agent systems and distributed architecture. We will also upgrade our searching technique by including semantic based searching technique along with keyword based searching technique using web ontology language to provide a more efficient web service searching experience to the service consumer and provider.

ACKNOWLEDGEMENTS

The authors would like to thank Subrat Gourab Bhadra, 4th year B. Tech student for his assistance in system implementation and experiments.

REFERENCES

[1] Sidnal N. S., Malashetty R. S. and Manvi S. S. 2010. Service discovery using software agents in Semantic Web. In: 11th International Conference on Control Automation Robotics and Vision (ICARCV). Singapore. pp. 139-143.

[2] Vadivel S. and Susila S. 2011. Agent based discovery of web service to enhance the quality of web service selection. International Journal of Computer Science and Network Security. 11(2): 159-163.

[3] Bin-hong X., Ying-jun Z. and Yong-yi G. 2010. A Web Service Matchmaker Based on Fuzzy Logic and OWL-S. In: International Conference on Computational Aspects of Social Networks (CASoN). Taiyuan. pp. 590-599.

[4] Chao K.-M., Younas M., Lo C.-C. and Tan T.-H. 2005. Fuzzy Matchmaking for Web Services. In AINA '05 Proceedings of the 19th International Conference on Advanced Information Networking and Applications. IEEE Computer Society Washington, DC, USA. 2: 721-726.

[5] Torres R., Astudillo H. and Salas R. 2011. Self-Adaptive Fuzzy QoS-Driven Web Service Discovery. In IEEE International Conference on Services Computing (SCC). Washington, DC. pp. 64-71.

[6] Wang P. 2009. QoS-aware web services selection with intuitionistic fuzzy set under consumer's vague perception. Expert Systems with Applications. 36(3): 4460-4466.

[7] Wooldridge M. and Jennings N. 1995. Intelligent Agents: Theory and Practice Knowledge Engineering Review. 10: 115-152.

[8] Hemayati M. S., Mohsenzadeh M., seyyedi M. a. and Yousefipour A. 2010. A framework for integrating web services and multi-agent systems. In: 2nd International Conference on Software Technology and Engineering (ICSTE). San Juan, PR. 2: V2-314-V312-319.

[9] Haibin Wang, Yan-Qing Zhang and Sunderraman R. 2006. Extensible soft semantic web services agent. Soft Computing. 10(11): 1021-1029.

[10] Zheng K. and Xiong H. 2012. Semantic Web service discovery method based on user preference and QoS. In: 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet). Yichang. pp. 3502-3506.

[11] Lopes A. L. and Botelho L. M. 2007. Executing semantic Web services with a context-aware service execution agent. In The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (SOCASE). Springer. 4504: 1-15.

[12] Ran S. 2003. A model for web services discovery with QoS. ACM SIGecom Exchanges. 4(1): 1-10.

[13] Chainbi W., Mezni H. and Ghedira K. 2010. PECoDiM: An Agent Based Framework for Autonomic Web Services. In: IEEE 6th World Congress on Services. Miami, Florida. pp. 543-550.

[14] Rajendran T. and Balasubramanie P. 2011. An Efficient Multi-Agent-Based Architecture for Web Service Registration and Discovery with QoS. European Journal of Scientific Research. 60(3): 439-450.

[15] Sapkota B., Roman D., Kruk S. R. and Fensel D. 2006. Distributed Web Service Discovery Architecture. In: International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications. p. 136.

[16] Nam S. and Kang Y. 2008. XML Schema Design for Web Service Quality Management. In: Second International Conference on Future Generation Communication and Networking. Hainan Island. Vol. 2.

[17] Sbodio M. L., Martin D. and Moulin C. 2010. Discovering Semantic Web services using SPARQL and intelligent agents. Web Semantics: Science, Services and Agents on the World Wide Web. 8(4): 310-328.

[18] Majithia S., Ali A. S., Rana O. F. and Walker D. W. 2004. Reputation-based semantic service discovery. In: 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 297-302.

[19] Badr Y., Abraham A., Biennier F. and Grosan C. 2008. Enhancing Web Service Selection by User Preferences of Non-functional Features. In: Proceedings of the 4th International Conference on Next Generation Web Services Practices. pp. 60-65. Seoul.

[20] Deng S., Wu Z., Wu J. and Li Y. 2008. An efficient two-phase service discovery mechanism. In: 17th international conference on World Wide Web. pp. 1189-1190.

[21] Gu T., Pung H. K. and Yao J. K. 2005. Towards a flexible service discovery. Journal of Network and Computer Applications. 28(3): 233-248.

[22] Siala F. and Ghedira K. 2011. A Multi-Agent selection of Web Service providers driven by composite QoS. In: IEEE Symposium on Computers and Communications. Kerkyra. pp. 55-60.

[23] Yu H. Q. and Reiff-Marganiec S. 2009. A Backwards Composition Context Based Service Selection Approach for Service Composition. In: IEEE International Conference on Services Computing. Bangalore. pp. 419- 426.

[24] Zeng L., Benatallah B., Ngu A. H. H., Dumas M., Kalagnanam J. and Chang H. 2004. QoS-aware middleware for Web services composition IEEE Transactions on Software Engineering. 30: 311-327.

[25] Zeng L., Benatallah B., Dumas M., Kalagnanam J. and Sheng Q. Z. 2003. Quality driven web services composition. In: Proceedings of the 12th international conference on World Wide Web. pp. 411-421.

[26] Rosenberg F., Celikovic P., Michlmayr A., Leitner P. and Dustdar S. 2009. An End-to-End Approach for QoS-Aware Service Composition. In: IEEE International Enterprise Distributed Object Computing Conference. Auckland. pp. 151-160.

[27] Yang Z., Shang C., Liu Q. and Zhao C. 2010. A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm. Journal of Computational Information Systems. 6(8): 2617-2622.

[28] Canfora G., Penta M. D., Esposito R. and Villani M. L. 2005. An Approach for QoS-aware Service Composition based on Genetic Algorithms. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, ACM New York, NY, USA. pp. 1069-1075.

[29] Grossmann I. E. 2002. Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques. Optimization and Engineering. 3: 227-252.

[30] Ming C. and Zhen-wu W. 2007. An Approach for Web Services Composition Based on QoS and Discrete Particle Swarm Optimization. In: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. Qingdao. 2: 37-41.

[31] Patil N. and Gopal D. A. 2011. Comparative Study of Mechanisms for Web Service Discovery based on Centralized Approach Focusing on UDDI. International Journal of Computer Applications. 14(1): 28-31.

[32] Benaboud R., Maamri R. and Sahnoun Z. 2012. Semantic Web Service Discovery Based on Agents and Ontologies. International Journal of Innovation, Management and Technology. 3(4): 467-472.