An Optimized Business Service Directory for the ESB Platform in SOA

RajniMohana[1], Deepak Dahiya[2]

Dept. of Computer Science and Engineering, JUIT Waknaghat, Solan, H.P, India

[1]rajni.mohana@juit.ac.in, [2]deepak.dahiya@juit.ac.in

Abstract:

Service Oriented Architecture (SOA) helps toprovide value added services to the user with agility by following a publish-find-compose technique. Business Service Directory is one of the key components of Enterprise Service Bus and its implementation involves many challenges. One of the essential challenges is how to provide a suitable set of service candidates faster. This paper proposes a business service directory which employs ranking based, quality driven publishing and searching. The proposed business service directory is a fuzzy expert system which requires no human intervention. The implementation of optimized business service directory is composed of two steps, first the rule base is designed which contains auto generated rules to rank the web service. Fuzzy clustering and PSO is used to not only generate rules automatically using training dataset but also removing redundant and less effective rules. The next step is to design inference engine which triggers the rules in the rule base. The proposed Business service directory minimizes administrative overhead and increases usability by locating the best web service among the large number of functionally equivalent web services and it's also adaptive in nature.

Keyword: *SOA, Web services, Business Service Directory, Enterprise Service Bus Platform, Fuzzy expert system, PSO, Fuzzy c means clustering, Quality of web services*

1. Introduction

In the world of enterprise information, Service Oriented Architecture (SOA) [1] plays an important role in developing Business to Business (B2B)/ Business to Consumer (B2C) applications with agility. SOA is defined as a deployment infrastructure on which new applications can be built quickly and easily [2]. It helps to provide business agility by configuring entities (services, registries, contracts, and proxies) to maximize loose coupling and reuse. Application built on SOA requires interoperating software applications, running on heterogeneous platforms and/or frameworks, which can be achieved by using Enterprise Service Bus (ESB) Platform. ESB is a middleware which provides a means for business to communicate with each other and with clients. It is an infrastructure for SOA service connection and message connection without intimate knowledge of each other's IT systems [3].

To establish a connection between the service provider and service consumer of a business application, Service search and selection is to be performed in Business Service Directory (BSD). BSD is a directory maintained by the ESB to store all the details of the services which are published in a Zone as shown in fig 1. Selecting the best web service among the various functionally equivalent web services, published in BSD is a complex problem.

The goal of this paper is to present an end to end solution for service discovery by focusing on BSD. It proposes an Optimized business service directory whose architecture consists of a rule base and inference engine. The rule base consists of the rules to rank a web service based on Quality of Service (QoS) [4]. These rules are generated automatically by fuzzy clustering and

DOI : 10.5121/ijcnc.2012.4511

particle swarm optimization (PSO). The inference engine locates the best web service according to the requirements of the service requester, by matching the rank of the service request with that of the services published in the business service directory. The motivation is to minimize the administrative overhead and increase usability by locating the best web service among the large number of functionally equivalent web services, published in business service directory.

The remainder of this paper is organized as follows: Section 2 discusses about the literature survey and some background knowledge about Fuzzy Logic, PSO and Fuzzy Expert Systems. Section 3 shows the proposed Business Service Directory for the ESB platform, Section 4 presents the design of Business Service Directory, algorithm used to generate the rules and how to use these rule to design an inference engine. Section 5 presents the detailed architecture of the optimized service registry component. Section 6 presents the results and observations of the designed component and its evaluation over other techniques. Lastly Section 7 summarizes the conclusion and future work. References are listed in Section 8.

2. Related Study

Enterprise service bus is a middleware which provides dependable and scalable infrastructure that connects heterogeneous applications and mediates their interactions, and makes them broadly available as services for additional uses. Fig 1 shows SOA in ESB platform. It consists of service requesters, service providers, ESB gateway, ESB name space Directory, Business service directory.

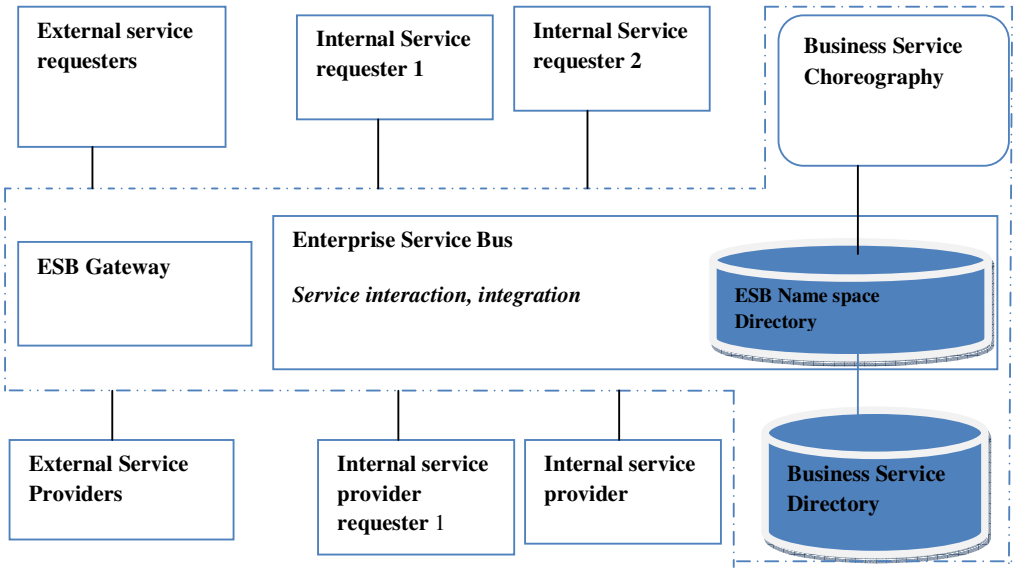

Fig 1: Enterprise Service Bus Platform in SOA [3]

Business Service Directory is design-time directories or customized service directories which give the details of the various services published by the service providers in the Zone [2]. When a service requester looks for a service in BSD a service discovery protocol is used to find out the best web service among various functionally equivalent services.

In literature survey we have analyzed various service discovery protocol presented by various researchers, they are listed and analyzed below:

Zhang etal[5] has designed a broker-based architecture called *QBroker*, to provide end-to-end QoS management for distributed services. Functionalities of QBroker include service discovery, planning, selection, and adaptation. The efficiency of QBroker is dominated by the running time of the service selection algorithm. They also designed efficient algorithms for quality-driven Web service compositions. Their model defines multiple QoS criteria and takes global constraints into account. It ensures that the selected services always meet the QoS requirements. They have also proposed heuristic algorithms to find near-optimal solutions in polynomial time which is more suitable for making runtime decisions. They have mapped the service selection to a 0-1 multidimensional multi choice knapsack problem (MMKP) [5].

VuongXuan Tran etal [6] have suggested an approach where the web services are selected based on QoS. The web services are ranked based on their quality attributes, but the issue raised by the researchers was that it's difficult to provide a precise value to the quality attributes of the web service. Thus they suggested using fuzzy logic to support using imprecise QoS constraints. The benefit of this approach is that a user does not need to mention crisp values of QoS properties. Instead the user can use fuzzy linguistic concepts to express their expectation of service quality.

However a user has to define at most as many rules as there are degrees of acceptance that s/he wants to differentiate. When a number of QoS properties are involved, the number of rules can be large and it becomes a tedious task for the user. It does not consider that some QoS criteria can be defined by using only crisp form such as criteria having Boolean or string value type [6].

Maolin Tang etal[7] quoted service discovery as so-called optimal web service selection problem. This paper proposes a new hybrid genetic algorithm for the optimal web service between some web service implementations like dependency and conflict constraints. When an implementation is selected for one web service, a particular implementation for another web service must be selected. This is so called *dependencyconstraint*. Sometimes when an implementation for one web service is selected, a set of implementations for another web service must be excluded in the web service composition. This is so called *conflict constraint*. Thus, the optimal web service selection is a typical constrained combinatorial optimization problem from the computational point of view. The hybrid genetic algorithm has been implemented and evaluated. They compared various techniques used for service discovery like penalty-based genetic algorithm, the repairing-based genetic algorithm and the hybrid genetic algorithm. It's shown that hybrid genetic algorithm is better than above mentioned techniques. The hybrid genetic algorithm is more suitable for those web service problems with a large number of abstract web services and a large number of constraints [7].

Al-Masri etal[8] has used Artificial Neural Networks (ANN) to classify the best web service according to the user requirements. The ANN classifies the web service based on QoS. They have designed a Web Service Crawler Engine (WSCE) that provides an active monitoring tool that continuously collects the most recent and up-to-date QoS values. They have also provided a QWS data set generated using WSCE for the purpose of research. They obtained most Web services from public sources on the Web, including UDDI registries, search engines, and service portals. The dataset consists of 5,000 Web services, each with a set of nine QoS attributes that we measured using commercial benchmark tools [8].

While analyzing the above protocols, it was observed that they involve soft computing techniques like *Fuzzy Logic, Genetic Algorithm (GA), and Artificial NeuralNetwork (ANN)* to discover the best web service based on QoS. Their advantages and disadvantages are discussed in Table 1.

Soft computing technique used in non-semantic approach	Advantages	Non - advantages
Neural networks	• High accuracy for float values	• Easy to over fit • Rules hard to extract and hard to understand
GA	• It updates the population and search for the optimum with random techniques.	• It is difficult to implement because of crossover and mutation. • System doesn't guarantee success
Fuzzy	• User does not need to specify concrete values of properties.	• a user has to define at most as many rules as there are degrees of acceptance that s/he wants to differentiate

Table 1: List of advantages and disadvantages of various soft computing techniques

Our research work involves PSO, because of its advantages over other soft computing techniques.

The advantages of PSO are:

- Easy to represent the interaction between attributes
- Consider several attributes once
- PSO is easy to implement and there are few parameters to adjust.
- All the particles tend to converge to the best solution quickly even in the local version in most cases.

Subsequent research work [6] using fuzzy approach states that the problem while using only fuzzy a user has to define at most as many rules as there are degrees of acceptance that s/he wants to differentiate. When a number of QoS properties are involved, the number of rules can be large and it becomes a tedious task for the user. Additionally, for each QoS property, there are number of membership functions being modeled but they may not satisfy some users.

We have proposed a different technique for service discovery in BSD which involves ranking the web services based on QoS using fuzzy clustering and particle swarm optimization (PSO) [9]. The significance of our approach is that it solves the above mentioned issues by generating rules automatically by using a training dataset QWS [10], thus designing the business service directory as a fuzzy expert system; The number of rules are also reduced by removing the rules having zero weight age according to the dataset. It uses fuzzy logic as it's applied in the problem domain where we have relaxing parameters and the results have to be computed exactly. It's used to compute overall QoS from the value of various QoS parameters. Fuzzy clustering is used to group the data according to the various linguistic values of the variables.PSO is used to optimize the rules according to the training data; it also reduces the redundant rules having zero effect in the system, thus reducing the time to trigger the rules.

The following section presents a brief introduction of fuzzy logic, fuzzy expert system, fuzzy clustering and optimization technique PSO. These concepts are used later in the paper to implement optimized service discovery protocol in service registry.

2.1 Fuzzy Logic

Applying fuzzy logic is beneficial for defining QoS description and measuring quality parameters in order to compute the overall QoS. Inference methods are used when the input-output relation can be expressed in the form of if-then rules. In this paper, we take advantage of the fuzzy logic for measuring overall QoS.

Aristotle [11] gave the theory of Boolean logic, which is two valued logic: true and false. Lofti Zadeh extended it to handle the concept of partial truth by presenting Fuzzy Logic, where the truth value may range between completely true and completely false [12]. It states that the variable values can be represented by degrees representing its closeness to truth. These types of variables are called as *linguistic variables* for example temperature, humidity etc. The values of the linguistic variables called as *linguistic values* are represented in the form of degrees for example temperature can be high, low or medium. These degrees may be managed by specific functions called as *membership functions. The membership function* is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. There are many types of membership function [12] but we will be using Z, S and Triangular in our paper as shown in fig 2.

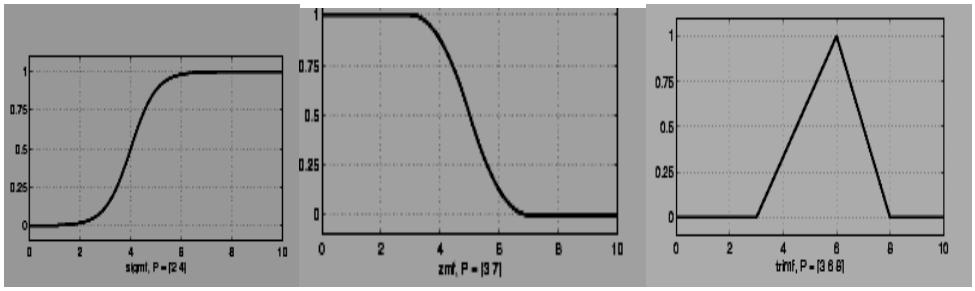

Fig 2: S membership function, Z membership function and Triangular membership function respectively

The rules in the fuzzy rule base use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified by the defuzzification interface into a crisp output which drives the system as shown in fig 6. This system is called as *fuzzy expert system [13].*

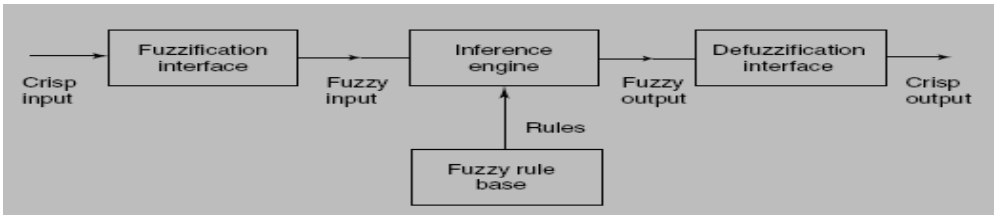

Fig 3: Fuzzy expert system

A fuzzy expert system [13] is application software that performs a task that would be performed by a human expert. It simply uses a collection of fuzzy membership functions and rules, instead of Boolean logic, to reason about data.

2.2Fuzzy C- means Clustering

Clustering is a mathematical tool that attempts to discover structures or certain patterns in a data set, where the objects inside each cluster show a certain degree of similarity. In fuzzy clustering, data elements can belong to more than one cluster, and associated with each element is a set of membership levels. These indicate the strength of the association between that data element and a particular cluster. Fuzzy c-means (FCM) [14] clustering is a process of assigning these membership levels, and then using them to assign data elements to one or more clusters. This technique was originally introduced by Jim Bezdek in 1981.FCM is a data clustering technique wherein each data point belongs to a cluster to some degree that is specified by a membership grade as an improvement on earlier clustering methods. It provides a method that shows how to group data points that populate some multidimensional space into a specific number of different clusters.

Areas of application of fuzzy cluster analysis include for example data analysis, pattern recognition, and image segmentation. The detection of special geometrical shapes like circles and ellipses can be achieved by so-called shell clustering algorithms. Fuzzy clustering belongs to the group of soft computing techniques (which include neural nets, fuzzy systems, and genetic algorithms) [14].

2.3 Particle Swarm Optimization

QoS parameters are often changing due to dynamic and volatile service environment. In such environment, Web Services need to be able to adapt dynamically trying to respect the service interaction. Changes are required to be captured; evaluated and proper actions need to be taken accordingly. PSO is used to optimize the rules according to the training data as explained in the algorithm 1.

PSO came into origin from the research of food hunting behaviors of birds. Researchers found that in the course of flight flocks of birds would always suddenly change direction, scatter and gather [9]. It is used to solve a wide array of different optimization problems. Their behaviors are Unpredictable but always consistent as a whole, with individuals keeping the most suitable distance. Through the research of the behaviors of similar biological communities, it is found that there exists a social information sharing mechanism in biological communities.

Each swarm of PSO can be considered as a point in the solution space. If the scale of swarm is N, then the position of the i-th (i=0,1, 2. ..N) Particle is expressed as Xi. The "best" position passed by the particle is expressed as pBest [i]. The speed is expressed with Vi. The index of the position of the "best" particle of the swarm is expressed with g. Therefore, swarm i will update its own speed and position according to the following equations [9].

$$Vi=w* V_i +c1 *rand ( ) *(pBest[i] *Xi)+c2*Rand)*(pBest[g] -Xi) \quad (1)$$

$$Xi =Xi+ V_i \quad (2)$$

Where c1 and c2 are two positive constants, rand ( ) and Rand ( ) are two random numbers within the range [0, 1], and w is the inertia weight, i refers to the swarm.

In other words, proposed BSD will lead to the design of a fuzzy expert system which will automatically perform service lookup. Fuzzy expert system consists of four components as shown in fig 3. The four components are fuzzification interface, inference engine, defuzzification interface and rule base. So optimized service registry component should also have all these four components to behave as a fuzzy expert system.

3. Proposed Business Service Directory for the ESB Platform.

Architecture of proposed Business Service Directory for the ESB platform is shown in fig 4. The proposed Business Service Directory is a fuzzy expert system. The fuzzification interface converts crisp input into fuzzy input, which is passed through inference engine. The inference engine ranks the published web service based on the rules in the Rule base. The rule base consists of automatically generated rules (using training data) by fuzzy clustering and particle swarm optimization (PSO) [7]. The defuzzification interface defuzzifies the fuzzy output into crisp output. The rules make the system automatic. These rules can be chosen by the human expert but our approach generates them automatically from the training dataset, thus it requires no human intervention in writing the rules for the system.

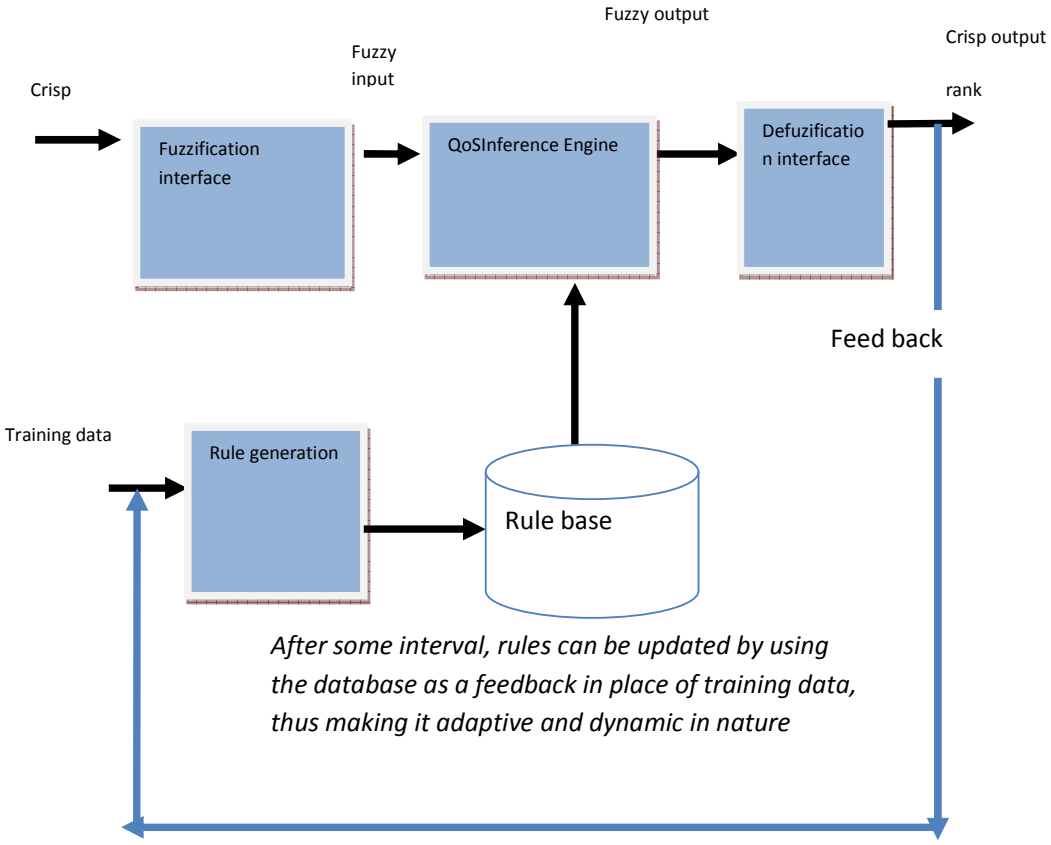

Fig 4: Architecture of optimized service registry

The service environment is dynamic and volatile, which leads to changes in their respective QoS parameters. In such an environment, web services needs to be able to adapt dynamically trying to respect the service interaction. Changes are required to be captured; evaluated and proper actions needs to be taken accordingly. Thus the rules can be updated by using the database as a feedback in place of training data, which makes it adaptive and dynamic in nature

The Proposed Business Service Directory is presented by means of UML class diagram in fig 5.

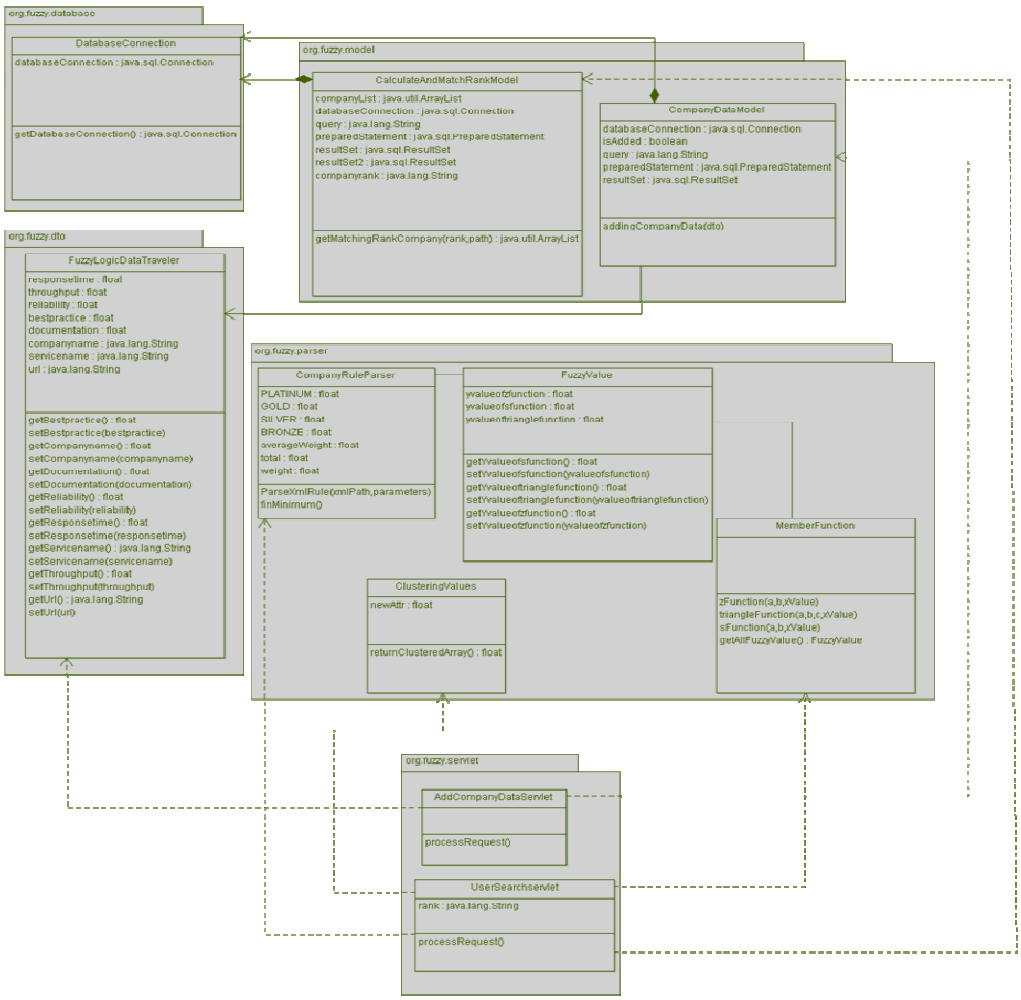

Fig 5. UML diagram of proposed Business service registry

The working of the proposed Business service directory can be very well explained using the information flow.

3.1 Information Flow for the Proposed Business Service Directory

The information flow for the proposed business service directory is described below:

1. If the user is a service provider who wants to publish his service in the BSD will require to do following steps:

    a. The service publisher will fill the form of input parameters which are quality attributes of the web services and submit it to business service directory
    b. In the business service directory, the input are accepted and ranked according to the rules by following a process of inference and defuzzification.
    c. The input details of service publishing form and service rank is then stored in the database i.e., business service directory in a WSDL format

2. If the user is a service requester

a. The service requester will fill the form of input parameters which are quality attributes of the web services and submit it to business service directory
b. In the business service directory, the input are accepted and ranked according to the rules by following a process of inference and defuzzification in decision making engine.
c. The business service directory is then searched for the web services with equal rank and provides the same service
d. The details of all the matching web service which were stored in the database by the service publisher is provided in the database

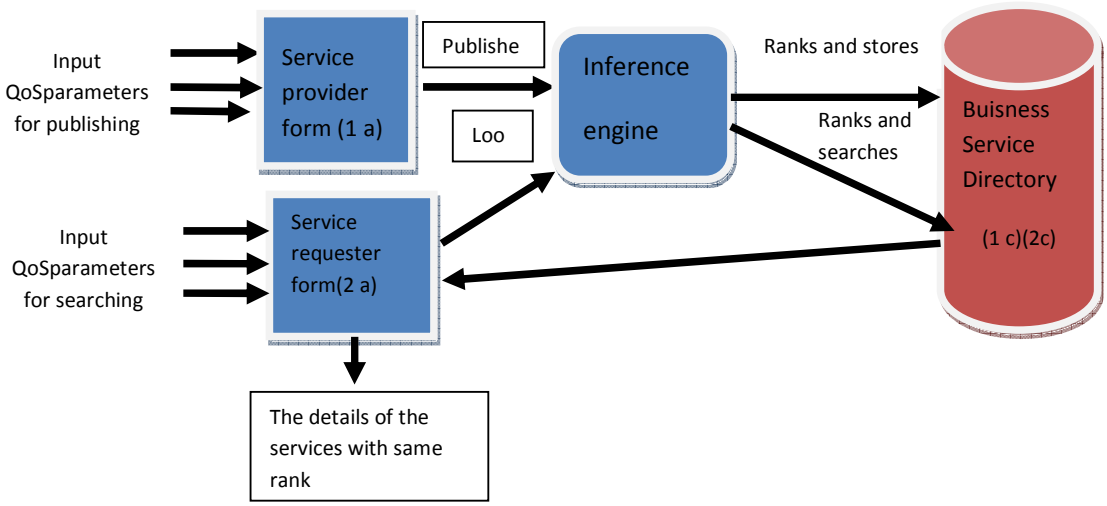

Fig 6: Information flow for optimized service registry component

4. Detailed Design for the Business Service Directory

The Designing of Business Service Directory presented in fig 6, encompasses two steps.

a. Generating optimized rules using the training dataset for performing service discovery in the Proposed Business Service Directory.
b. Designing the inference engine using a java EE platform, which triggers the rules to rank and match the request with the rank of the service published by the service provider.

4.1 Automatic generation of rules from dataset

To rank the web services, the rules can be provided by the expert in the domain or alternatively it can be generated using the algorithm given below. The rules are generated using a dataset. The dataset used is called as training set. The technique given below also generates lesser number of rules i.e., it removes all the rules which are having zero impact on the system.

The algorithm is described below:

The above problem domain which involves calculating the rank of a web service based on the values of QoS parameters can be considered as TSK model. The shape of membership functions for the input variables are Z, S and Triangular respectively. The output is constant with parameters ranging from 0 ->1. The notations used in the algorithm are described as under

Population is the no of swarms

Pbest is the best value obtained in each iteration
Gbest is the best value obtained among all the pbest

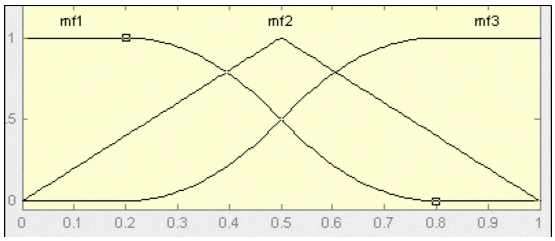

Fig 7. Membership functions for input QoS Parameters.

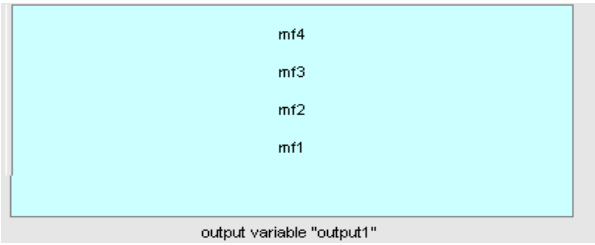

Fig 8.Membership functions for output QoS Parameters.

The algorithm for fuzzy rule based model is

1. Assign membership function to all the input variables .If there are two linguistic values of the variables then assign S to the lower on and Z to the higher one. If the no of values are more than two assign triangular membership functions to all the vales lying between higher and lower. The parameters for S= [0.2, 0.8]

Z= [0.2, 0.8]

Triangular = [0, 0.5, 1]

As shown in fig 7

2. Assign membership values to the output variables If values <=2 assign parameters 0 to low and 1 to high respectively. If values > 2 assign parameters 0.5 to all the intermediate values as shown in fig 8.

3. Calculate the max and min of each input parameter and output parameter from the data set used to train the particles.

4. Using FCM technique of fuzzy clustering assign the center points to the to membership function of input and output variables

5. Create a fuzzy inference engine containing all the possible combination of rules formed from input variables

Initialize the variables used in equation 1 and 2

i.e., Population size= 10

Maximum iteration = 100

wmax = 0.9

wmin = 0.3

c1 =2

c2 = 2

6. Assign randomly an output to the rule set and optimize the result and calculate the deviation from the result called error.

7. Pbest=result

8. Pbestfitness= error

9. Gbest= Minerror

10. velocity V is also assigned a random value

11. Vmax = 1

12. Repeat steps from 12 – 18 till iteration <maxiteration& flag == 0(flag checks whether the output is in acceptable limit.

$$VI=w*VI+cl*randO*(pBest[i]-Xi)+c2*RandO*(pBest[g]-Xi$$

$$Xi=Xi+Vi$$

X is considered as the result of the rule set which is optimized by the particles.

13. Calculate the output according to the input in the training set.

14. Match the output; calculate the deviation from the result

15. pbest=X

Pbestfitness= error

Gbest=pbest

GbestFitness= min of gbest

16. if the deviations is in acceptable limit mark flag=1

17. Assign the value of gbest as the consequent of the rules.

18. Weight /degree of each rule is equal to the maximum value of membership value of the input variables

In order to train the particles publicly available Quality of Web service (QWS) dataset [15][16] is used. The main goal of this dataset is to offer a basis for Web service researchers. This

dataset is collected considering a subset of 365 real web service implementations that exist on the Web today. The services were collected using our Web service Crawler Engine (WSCE) [15]. The majority of Web services were obtained from public sources on the Web including Universal Description, Discovery, and Integration (UDDI) registries, search engines, and service portals. The public dataset consists of 365 Web services each with a set of nine Qualities of Web service (QWS) attributes that we have been measured using commercial benchmark tools. Each service was tested over a ten-minute period for three consecutive days by the researchers [12].

The various quality attributes considered are shown in table 2:

Parameter Name	Description	units
Response Time	Time taken to send a request and receive a response	ms
Availability	Number of successful invocations/total invocations	%
Throughput	Total Number of invocations for a given period of time	invokes/second
Successability	Number of response / number of request messages	%
Reliability	Ratio of the number of error messages to total messages	%
Compliance	The extent to which a WSDL document follows WSDL specification	%
Best Practices	The extent to which a Web service follows WS-I Basic Profile	%
Latency	Time taken for the server to process a given request	ms
Documentation	Measure of documentation (i.e. description tags) in WSDL	%

Table 2: Web service quality attributes [10]

Out of the above mentioned quality attributes a data set of five quality attributes based on the availability in QWS dataset was considered as input variables. In the research work described in this paper, the dataset for *web service phone* was considered

I. Response Time (ms) ={high , average, low}
II. Throughput (hits/sec) ={ high , average, low}
III. Reliability (%) ={ high , average, low}
IV. Best Practices (%) ={ high , average, low}
V. Documentation ={ high , average, low}

One parameter is considered as output variable

I. Rank ={ Platinum (High quality) ,Gold ,Silver ,Bronze (Low quality)}

The range for linguistic values (high, average and low) of QoS is calculated by performing fuzzy clustering on the training data. A data set which was generated using the demo [10] and searching for a *web service phone* is shown in table 3.

Name	Response Time (ms)	Throughput (hits/sec)	Reliability (%)	Best Practices (%)	Documentation (%)	Rank
DOTSGeoPhone	126.2	12.3	78.7	80	86	Platinum
Phone	150.45	7.4	82.1	82	37	Gold
DOTSPhoneAppend	118.5	0.7	70.2	80	90	Gold
PhoneVerify	131	1.6	65.9	72	41	Gold
PhoneNotify	437.62	1	68.4	69	93	Silver
PhoneService	133	1.4	64.7	82	10	Bronze
Phonebook	464	3.1	43.2	80	2	Bronze

Table 3. Data set on searching the web service phone [10]

4.2 Designing the Inference Engine

The inference engine uses the above generated rules to rank the web services published by the service provider. It will then save the details of the web service in a database. If a request arrives from a service requester, it will be also ranked by the inference engine and based on the criteria of the service name and rank, the request will be searched. If a match found the details as well as the web address of the web service is provided to the service requester. The service requester can then directly communicate with the web service and establish the connection.

Architecture of Inference Engine

The registry is implemented using tools and technologies comprising of NetBeans IDE, JSP, Java EE, MySQL, Apache Tomcat, and XML. Servlet is used to process user request and JSPs is used to create the view which are the core component of the project. This project follows Model View Controller (MVC) architecture which keeps the code clean and manageable. Through MVC the processing, view and database code are kept separate.

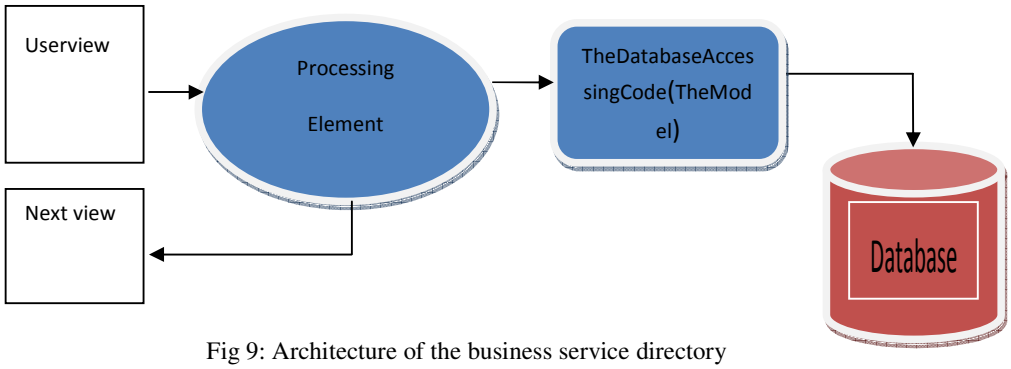

Fig 9: Architecture of the business service directory

The User view will constitute of a fuzzy rule parser. All the rules are stored in an XML file. Fuzzy rule parser triggers the rules when a request or a publishing takes place. XML is chosen for database accessing code because it requires less time than database hit and it's easy to manipulate. XML has an advantage that a large number of data can be stored and can be manipulated easily. Defuzzification is done using weighted average technique.

The registry provides a publishing API. The web service publishers can publish their web services on the basis of some parameter. Initially all the given parameters are stored in a database so that there should not be any delay in processing. Later all the data are converted into XML, as searching in the database consumes more time than searching in the tree structure of XML. Database chosen is MySQL.

The enhanced entity relationship (EER) diagram in fig 10 describes the relationship and attributes in the various tables of database.

Fig 10: EER diagram of used database

5. Implementation of the Optimized Business Service Directory

The fuzzy expert business service directory is implemented in two phases:

a. Generating the rules by implementing the algorithm described in section 4.1
b. Implementing the inference engine described in section 4.2

First the rules are generated using the QWS dataset and then an inference engine is developed for the optimized business service directory which has a service requester and provider form. The data when fed on the form will process according to the rules using the designed fuzzy rule parser. The output of the rule is then defuzzified to give a crisp output. The crisp output is the rank of the web service. If the rank is calculated for the service provider, the application will store all the details of the web service along with the rank. When the service requester is looking for a web service in the database, his requirement is also ranked by the above mentioned technique, and then the web service with the specified rank is located in the database. If the search is successful, then the details of the web service stored in the database are provided to the service requester.

5.1 Automatic Generation of Rules

Rules generated after implementing the algorithm explained in section 4.1 using Matlab 7.0 is shown in table 4. The number of rules generated is 26.

1.	if Response Time is low and throughput is low and reliability is average and best practices is low and documentation if low then output is bronze 0.121455
2.	if Response Time is low and throughput is low and reliability is average and best practices is low and documentation if average then output is silver 0.467807
3.	if Response Time is low and throughput is low and reliability is average and best practices is average and documentation if low then output is silver 0.034978
4.	if Response Time is low and throughput is low and reliability is average and best practices is average and documentation if average then output is silver 0.134724
5.	if Response Time is low and throughput is low and reliability is average and best practices is average and documentation if high then output is platinum 0.000811
6.	if Response Time is low and throughput is low and reliability is average and best practices is high and documentation if low then output is platinum 0.847967
7.	if Response Time is low and throughput is low and reliability is average and best practices is high and documentation if high then output is gold 0.439231
8.	……
9.	……..
10.	……
23.	if Response Time is high and throughput is low and reliability is average and best practices is low and documentation if high then output is platinum 0.573544
24.	if Response Time is high and throughput is low and reliability is high and best practices is low and documentation if high then output is silver 0.426456
25.	if Response Time is high and throughput is average and reliability is low and best practices is average and documentation if low then output is platinum 0.000285
26.	if Response Time is high and throughput is average and reliability is low and best practices is high and documentation if low then output is platinum 0.154625

Table 4: Rule generated from the data set and there corresponding weight

The number of rules generated (26) using the rule based model described in section 4.1 are less in comparison to generated by using Fuzzy Logic i.e., 243. Lesser the number of rules to trigger lesser is the seek time required to locate the best web service.The rules given above are stored in the rule base. Whenever a service request comes from a service requester, it places his requirements, for example if a phone service which is highly reliable and has average response time is requested. The inference engine will rank the request according to the rules given in the rule base and match it with the rank of services registered by their service providers. Once a

match is found the service requester will be given the network address of the selected service provider.

5.2 Implementation of Inference Engine

The designed Business Service Directory consists of the inference engine using a Java EE platform. The controller shown in fig. 9 consists of a fuzzy rule Parser. As we know Directory provides web service on the basis of rank. The rank is calculated on the basis of some parameters e.g. response time, throughput etc., supplied by the web service publishers.

The rules are as under in an XML format:

```
<!ELEMENT response-time EMPTY>
<!ELEMENT throughput EMPTY>
<!ELEMENT reliability EMPTY>
<!ELEMENTbestpractice EMPTY>
<!ELEMENT documentation EMPTY>
<!ELEMENT rank EMPTY>
<!ATTLIST response-time value (high | low | average) #REQUIRED>
<!ATTLIST throughput value (high | low | average) #REQUIRED>
<!ATTLIST reliability value (high | low | average) #REQUIRED>
<!ATTLISTbestpractice value (high | low | average) #REQUIRED>
<!ATTLIST documentation value (high | low | average) #REQUIRED>
<!ATTLIST rank value ( platinum | gold | silver | bronze) #REQUIRED>
<!ELEMENTrule(responsetime,throughput,reliability,bestpractice,documentation,rank)>
<!ELEMENT rules (rule+)>
```

To parse the XML a DOM parser is used. DOM parser provides accessing of XML element in a hierarchical way. This makes accessing easier and faster. All XML elements are loaded simultaneously in memory and can be accessed in any order.

5.3 Working of the Proposed Business Service Directory

The tools used for implementing the component comprised of NetBeans IDE, JSP, Java EE, MySQL, Apache Tomcat and XML. This section presents the snapshots of the implemented components:

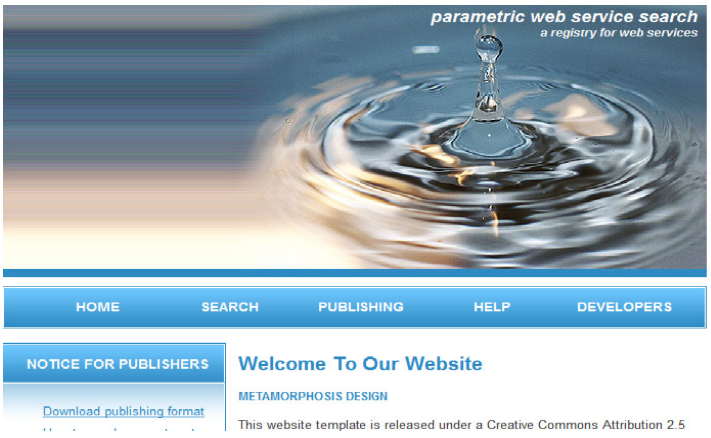

Fig 11: Snap shot of the home page

When clicked on publishing icon by the service provider the following form appears. The provider will have to fill the details of the publishable web service. After entering the details in the form the details are entered in the database as shown in fig 14.

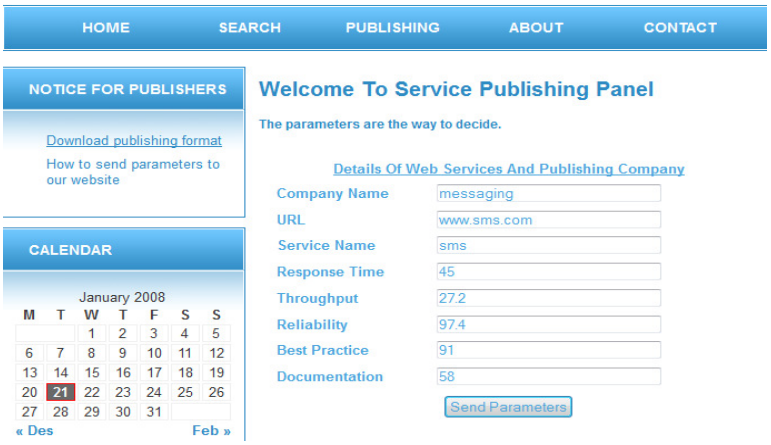

Fig 12: snapshot of the service publishing panel

A message appears that the web service and its details are added in the database as shown in fig 13.

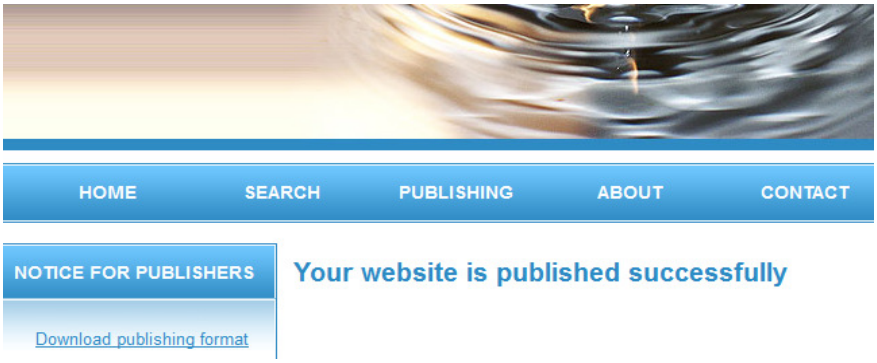

Fig 13: snapshot of the submission of a web service in the service publishing panel

If a service requester searches the web service, he will click on the search icon on the home page. After entering the details of the web service the user is looking for, the list of relevant web service is provided as shown in fig 14 and 15 respectively.

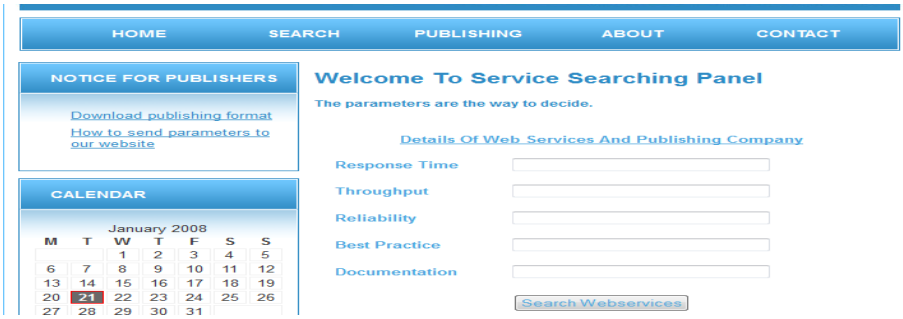

Fig 14: snapshot of the submission of a web service in the service searching panel

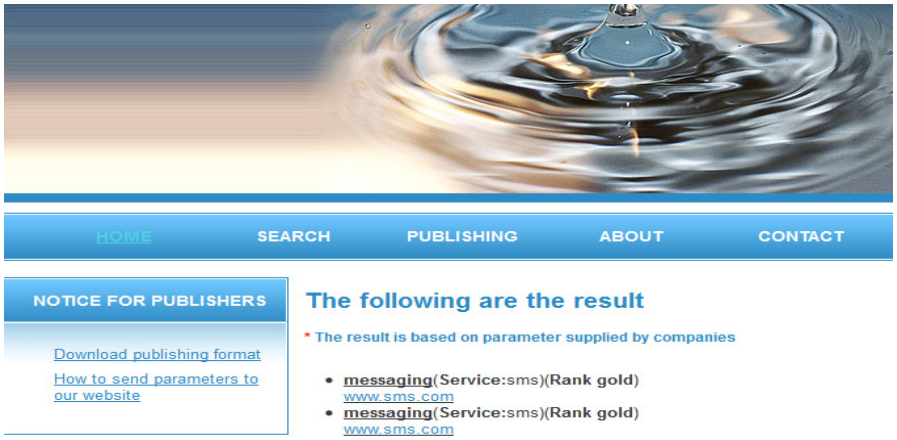

Fig 15: snapshot of the submission of a web service in the service publishing panel

The website also provides an API which can be included in the web services of service provider and requester so that they can access the form directly, without opening.

6. Results and Observations

After implementing the optimized business service directory, a check was done on the amount of heap memory consumed by the component and the time required to publish and search a service. The below fig shows the heap memory consumption of the component:

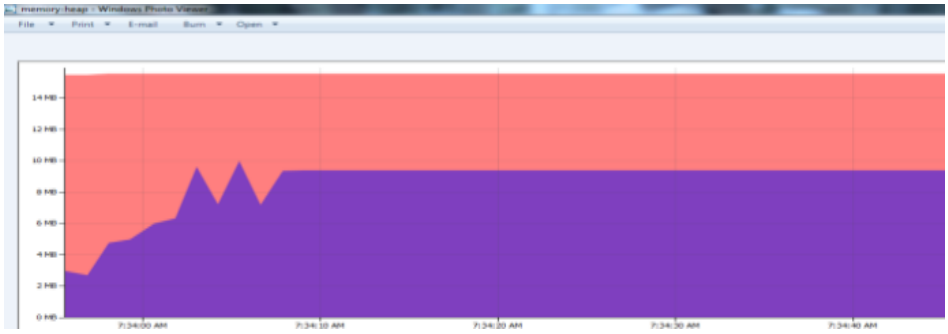

Fig 15: snapshot of the heap memory consumption of the component

It can be seen that there is fluctuation in the heap memory consumed when the process in NetBeans IDE starts execution and later it depicts a constant consumption of memory i.e. in this case it consumes 10MB of memory.

The next fig shows the time required to perform web publishing. The time required to process the request to publish the service is $122*10^{-3}$ sec.

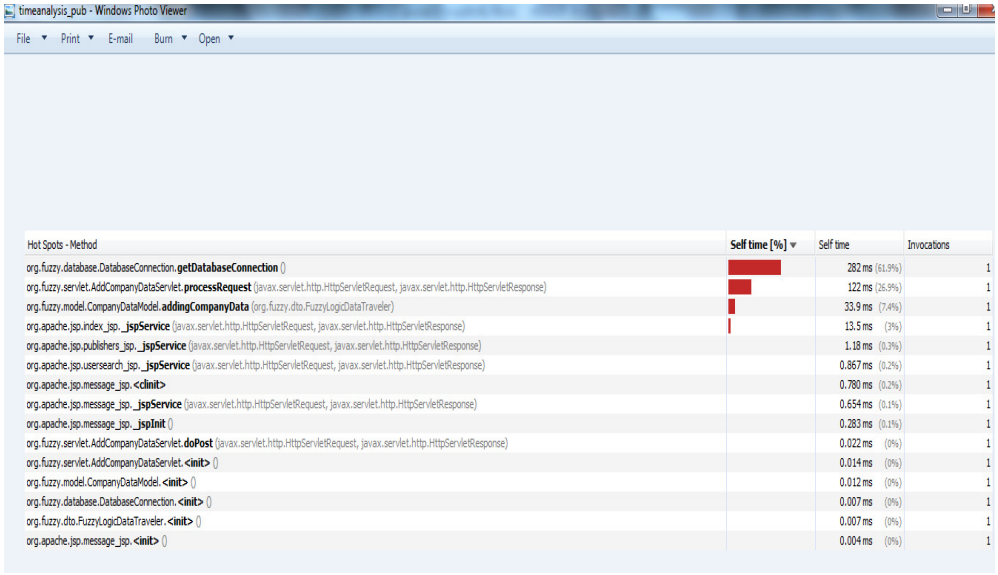

Fig 16: snapshot of the time consumed to perform web publishing

The next snapshot shows the time required to process the request of web search is$171*10^{-3}$sec, and to parse the xml rules time required is $39.9*10^{-3}$ sec.

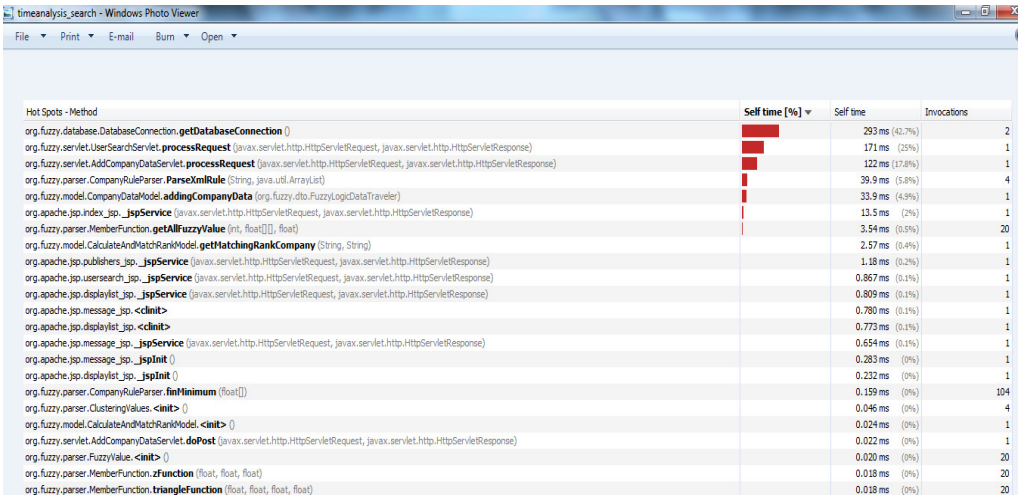

Fig 17: snapshot of the time consumed to perform web searching

On carrying out the qualitative analysis of the techniquedescribed aboveto generate the rules, as a part of the research work in comparison to the technique which uses fuzzy logic the following observations are made:

1. The time required to search the web service in the database also called as seek time will be significantly less as it will be dependent on the number of rules required to trigger and the number of rules is less in the proposed technique as compared to implementing the BSD using fuzzy logic. The time complexity to parse the rules is O (N) where N is the no of rules. Lesser the number of rules to trigger less will be the seek time. The seek time is also affected by the database size and the number of quality attributes considered qualifying the web service.

In the scenario considered and the related work described in the paper, efforts were directed towards search for seven web services and it was found that the time required by the rule generation technique using PSO and fuzzy clustering is faster than the alternative technique which uses fuzzy logic [4]. The time was calculated by considering the time required to execute *ParseXmlRule(String xmlPath, ArrayList parameters)* function in the project.

services	**Service 1**	**Service 2**	**Service 3**	**Service 4**	**Service 5**	**Service 6**	**Service 7**
Time required to parse the 26 rules generated by Fuzzy clustering and PSO	$187*10^{-3}$ sec	$269*10^{-3}$ sec	$39.9*10^{-3}$ sec	$222*10^{-3}$ sec	$301*10^{-3}$ sec	$322*10^{-3}$ sec	$343*10^{-3}$ sec
Time required to parse the 243 rules generated by Fuzzy logic	$693*10^{-3}$ sec	$613*10^{-3}$ sec	$309*10^{-3}$ sec	$371*10^{-3}$ sec	$480*10^{-3}$ sec	$530*10^{-3}$ sec	$574*10^{-3}$ sec

Table 5: Comparison of time consumed to parse 26 rules vs. 243 rules

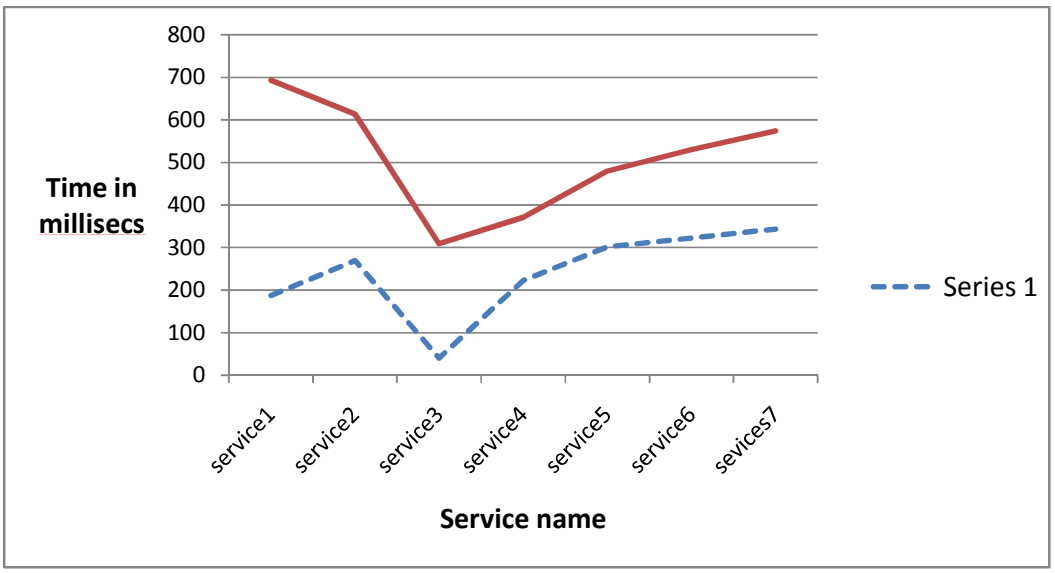

Fig 18: Graph representation of time required to fire 243 vs. 26 rules

2. The rules are successfully generated automatically using dataset thus making the system intelligent, in comparison to the technique using fuzzy logic [6] where the rules are to be entered by a human expert. Human intervention makes the system error prone and manual.

3. The rules are adaptive i.e., any change in the dataset or the ranking criteria will automatically be reflected in the rules and thus a new set of rules will be generated. The web services can be ranked according to the new rules.

4. The quality of rules is dependent on the training dataset; the data should be less overlapping and should have all varieties of output. The rules can still be generated using less number of entries in the dataset.

7. Conclusion and Future Work

The proposed business service directory is automatic in nature as it a fuzzy expert system, which will rank the web services according to the rules generated by dataset. The PSO and fuzzy clustering reduces the rules.

Considering the available data set, the number of rules is reduced from 243 (product of linguistic values of input and output variables) to 26. The lesser the number of rules, faster will be the processing of ranking. The component is designed and can be used as a registry to publish and search the web services. The architecture is adaptive in nature as any change in QoS of a web service will change the rank of the web service. The proposed optimized service registry will enable one to develop a better B2B or a B2C kind of e-commerce application with agility. The service requester can compare among the list of web service and choose the appropriate service provider based on its requirements.

The future work will involve optimizing and integrating the elements of the ESB platform incrementally.

Finally to summarize, the proposed model is better as it automatically monitors the rank of the web service using the generated rules.

8. References

[1] Liamo'Brien, Paulo Merson, Len Bass"Quality Attributes for Service-Oriented Architectures" in the Proceedings of the International Workshop on Systems Development in SOA Environments SDSOA '07,doi:10.1109/SDSOA.2007.10

[2] Understanding SOA with Web Services by Newcomer published by Pearson Education India

[3] Patterns: Implementing an SOA Using an Enterprise Service Bus IBM Red book published in july 2004

[4]Liamo'Brien, Paulo Merson, Len Bass"Quality Attributes for Service-Oriented Architectures" in the Proceedings of the International Workshop on Systems Development in SOA Environments SDSOA '07,doi:10.1109/SDSOA.2007.10

[5] Yu, T., Zhang, Y., and Lin, K.-J. 2007. Efficient algorithms for Web services selection with end-to-end QoS constraints. ACM Trans. Web 1, 1, Article 6 (May 2007), 26 pages. DOI = 10.1145/1232722.1232728

[6] VuongXuan Tran; Tsuji, H.*QoS based Ranking for Web Services: Fuzzy Approaches* In proc. of the 4th International Conference on Next Generation Web Services Practices, Seoul, pp 77 – 82, 20-22 Oct. 2008

[7] MaolinTang ,Lifeng Ai *; A Hybrid Genetic Algorithm for the Optimal Constrained Web Service Selection Problem in Web Service Composition* In proc. Of IEEE Congress ontheEvolutionary Computation (CEC), 2010 pp 1 – 8 18-23 July 2010

[8] Al-Masri, E.; Mahmoud, Q.H.; *Discovering the Best Web Service: A Neural Network-based Solution* In proc. of the IEEE International Conference on Systems, Man and Cybernetics, 2009. San Antonio, TX, USA, pp 4250 – 4255 11-14 Oct. 2009

[9] Kennedy, J.; Eberhart, R.; Particle swarm optimization. Proceedings. In proc. of the IEEE International Conference on Neural Networks, 1995, Perth, WA , Australia vol.4

pp: 1942 - 1948

[10] http://www.uoguelph.ca/~qmahmoud/qws/index.html as on jan 2011

[11] http://en.wikipedia.org/wiki/Fuzzy_logic as on july 2011

[12] http://www.mathworks.com/help/toolbox/fuzzy/fp754.html as on july 2011

[13] http://www.austinlinks.com/Fuzzy/expert-systems.html as on july 2011

[14]http://en.wikipedia.org/wiki/Cluster_analysis on july 2011

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

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