# IWSS: Intelligent Web Service Selection for Service-Oriented Applications

A dissertation Submitted by

**Rabia Bashir**

**(2010-NUST-MS PhD-CSE-29)**

Submitted to the Department of Computer Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Software Engineering

**Supervised By:**

**Dr. Aasia Khanum**

**College of Electrical & Mechanical Engineering**

**National University of Sciences and Technology**

**2012**

**THE COMMITTEE**

Approved as to style and content by:

**Dr. Aasia Khanum ,** *Supervisor*

**Dr. Farooque Azam,** *Member*

**Dr. Saad Rehman,** *Member*

**Dr. Arslan Shaukat,** *Member*

**Dr. Aasia Khanum**

Assistant Professor, Department of Computer Engineering

# DEDICATIONS

*“Dedicated to my parents and family who have supported me all the way throughout the course of my life”*

# ACKNOWLEDGEMENTS

In the name of Allah, the Most Beneficent and Most Merciful Who bestowed me with audacity for completing this thesis. Foremost, I am really indebted to my supervisor Dr.Aasia Khanum for her motivation, continuous support and colossal knowledge. Her guidance throughout the experimental and thesis work made it possible to successfully complete this research. I am very thankful for her enlightened supervision, inspiring guidance, encouragement and constructive suggestions throughout the research work. Besides my supervisor, I would like to express my gratitude to rest of my thesis committee: **Dr. Farooque Azam, Dr.Saad Rehman and Dr.Arslan Shaukat** for their insightful comments and suggestions.

Last but not least, I owe deepest gratitude to my beloved parents and family for their continuous support and prayers for the accomplishment of this task.

# ABSTRACT

The world of web services is gigantic and the demand of web services is increasing day by day because of functionalities provided by them. With increase in number of available web services the task of discovery and selection of web services is becoming exigent when there is large number of services providing the similar functionality.

Mainstream approaches for WS selection rely on functional description of the service as published by the service provider. Moreover, there is no provision for adapting the service selection behavior by taking into account user’s run-time experience with the service. The framework presented in this paper combines functional parameters of published WS with accumulated store of previous users’ Quality of Experience (QoE) in a Case Based Reasoning (CBR) system for precise selection of best WS for a user query. Proposed framework is also incorporating the users’ preferences with service request parameters and collects users’ feedback to select the most appropriate service for a user’s request. CBR remembers service selection made in presence of functional and QoE parameters to guide decision-making when faced with similar situation in future

Experimental results show that the proposed approach gives much more precise results than naive approaches. It provides the optimal service to users with ease of usage.

## Table of Contents

# Table of Contents

# Table of Contents

# List of Abbreviations

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **CBR** | Case Based Reasoning |
| **DQoS** | Decision Quality of Service |
| **DWS** | Delegation Web Service |
| **IWSSF** | Intelligent Web Service Selection Framework |
| **QoE** | Quality of Experience |
| **SOA** | Service Oriented Architecture |
| **SRMA** | Service Request Matching Algorithm |
| **SOAP** | Simple Object Access Protocol |
| **SLA** | Service Level Agreement |
| **SR** | Service Request |
| **SPD** | Service Preference Document |
| **SC** | Service Case |
| **UDDI** | Universal Description Discovery and Integration |
| **URI** | Uniform Resource Identifier |
| **URL** | Uniform Resource Locator |
| **WS** | Web Service |

# List of Abbreviations

| | |
|---|---|
| **WWW** | World Wide Web |
| **WSDM** | Web Service Distribution Management |
| **WSDL** | Web Service Description Language |
| **WSMF** | Web Service Modeling Framework. |
| **WSRF** | Web Service Resource Framework |
| **WSMO** | Web Service Management Ontology |

# List of Figures

# List of Figures

# List of Tables

Chapter 1

# 1 Introduction

In this chapter, we shall discuss the main concepts which are used in this thesis. Artificial Intelligence in Web Services (WS) is discussed in section 1.1. Section 1.2, 1.3 and 1.4 discusses the web services, emergence of Service Oriented Architecture and detail of service oriented architecture (SOA).Section 1.5 discusses Roles in SOA. Case Based Reasoning (CBR) system and CBR life cycle is discussed in section 1.6 and 1.7.Relationship of CBR with machine learning is discussed in 1.8. Problem statement is discussed in section 1.9 and section 1.10 provides the details of our contribution. Thesis organization is described in section 2.

## 1.1 Artificial Intelligence in Web Service Selection

According to Ray Kurzweil, *Artificial Intelligence (AI) is the defined as : "The art of making machines(computers) that carry out such tasks which need intelligence when they are done by human " [1].There are various branches of AI which include: computer vision, learning and expert system, problem solving and planning natural , robotics and language processing[2].*

Research in AI and WS is bringing together the solutions that will lead us towards mature and next generation of World Wide Web (WWW).Development of Web Service Modeling Framework (WSMF) is an evident example of involving AI in this domain [3].

## 1.2 Web Services (WS)

According to W3C, WS is defined *"as a software application which is recognized /identified by Uniform Resource Locator (URL) and artifacts of XML are used to define, describe and discover its bindings and user interfaces which also help direct communication with software applications by using messages in XML format through internet protocols"*[4].

Web Services are standardized software components that can inter-operate in a distributed manner using standardized protocols for communication. Being independent of hardware, platform (operating system) and programming languages, WS offer a large variety of benefits including inter-operability, reusability, deployability etc. WS present a new web model where different sites can share information dynamically. This exchange is most important for e-business where it provides an opening to accomplish business quickly and proficiently compared to traditional technologies [5].WS are the emerging technology to simplify large business-oriented operations and more software is being made available to customers as WS with easy-to-use interfaces [6].

## 1.3 An Emerging Technology of SOA and Web Services

Web Services is a technology commonly used to implement SOA which is architectural style. The purpose of SOA is to attain loose coupling between software agents. These agents interact with each others. Service is defined as unit of work which is performed by service provider to achieve the final results for service consumers. Software agents perform the roles and responsibilities of service provider and consumer [7].Though much work has been done on SOA and WS but still there is confusion about these two terminologies between software developers [8].

## 1.4 Architecture of SOA

Basic architecture of SOA as shown in figure 1 consists of Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Universal Description Discovery and Integration (UDDI) [9].

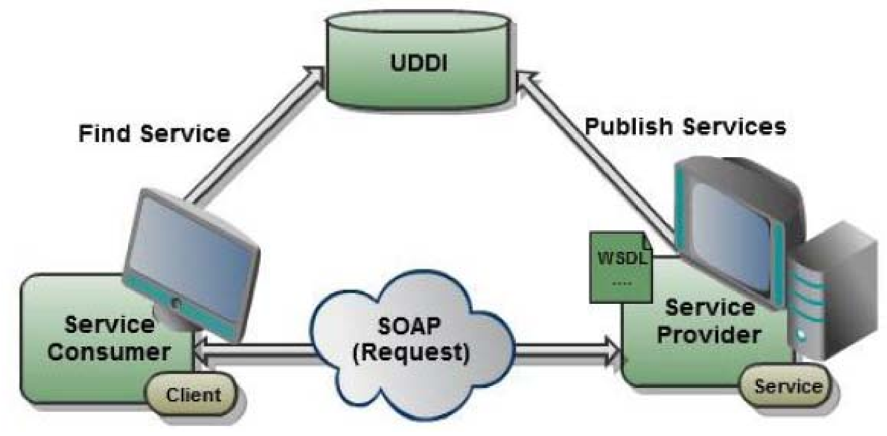


**Figure 1: Architecture of SOA**

### 1.4.1 Simple Object Access Protocol (SOAP)

SOAP [9] is a protocol that is XML based and is used for the data exchange by using Hyper Text Transfer Protocol (HTTP). For sending messages in XML format among software applications it provides standard, easy and simple methods. SOAP is being used by WS to exchange messages between service provider and consumer as all the services and web browsers provide support of

HTTP. SOAP messages are independent of platform and programming languages of applications among which they are passed. Basic structure of SOAP is shown in figure 2.

SOAP message consists of following elements:

#### 1.4.1.1 SOAP Envelope

It depicts that SOAP message is XML document and it has two parts Header and Body

#### 1.4.1.2 SOAP Header

It is optional and retains message related information, e.g., date of message when it was sent and authentication data etc. Header is considered as an information place holder and that information is not essentially depending on application. Header is mostly used to contain security and coordination information.

#### 1.4.1.3 SOAP Body

It contains actual XML formatted message data which is application specific.

#### 1.4.1.4 SOAP Fault

It is optional and holds information of server or client error. If SOAP message is not processed then fault is returned [9].

SOAP fault must contain the following information [10]:

- **Fault Code**: representing the error class and subcode.
- **Fault String**: Human can easily read the explanation of fault.
- **Fault Actor**: who is responsible for causing the fault?
- **Detail**: It is application specific data linked with the fault.

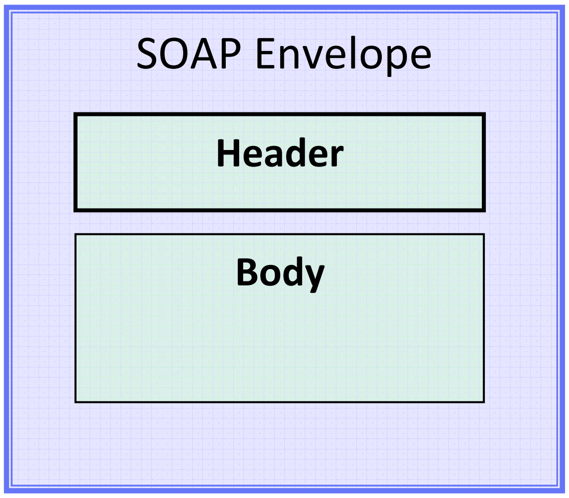


**Figure 2: Basic SOAP Structure**

### 1.4.2 Universal Description Discovery and Integration (UDDI)

UDDI is service registry or directory containing web services as all service providers publish their services in it. Service consumer search the service in UDDI as naming and service discovery system is essential because without it, it is much cumbersome to find a particular service for which user is searching for. Therefore, UDDI increases the awareness about web services that are currently available. Web service information of business organizations can be published in this directory but this provided information must be adequate enough to get access to these services in UDDI [11].

### 1.4.3 Web Service Description Languages (WSDL)

WSDL [11] document depicts web services and it is in XML format. It provides information about service location and operations that it performs. It provides service users with all the information that is needed them to interact with that service. WSDL helps service consumer to lean from where to access the service, which functions that service performs and the exact format for messages to send to that service. WSDL document acts as a contract among service provider and consumer. By holding this contract both are liable for data exchange in a standard way despite of any specific platform being used.

WSDL file consists of following elements:

#### 1.4.3.1 Binding

All the communication protocols which are used by the operations performed by the services are defined by binding element.

#### 1.4.3.2 Port

It defines binding address i.e. communication port.

#### 1.4.3.3 Port Type

It defines the functions performed by the web services via defined service interfaces.

#### 1.4.3.4 Types

It defines data types which are used by web services for message sending between client and server.

#### 1.4.3.5 Service

It specifies the address to access the web service.

## 1.5 Roles in SOA

There are three main roles in SOA [12].

### 1.5.1 Service Provider

It publishes or registers its services in UDDI and accepts the web service request of service consumer and executes these requests.

### 1.5.2 Service Consumer

It is software module /application or another service that calls for service. It searches for needed service in service registry, binds to that service and then executes the service operation. Service is executed by the consumer in accordance with interface contract.

### 1.5.3 Service Registry

It is service directory or repository which manages all the available service registered by service providers and allows consumers to search the web service provider interfaces [12].

## 1.6 Case Based Reasoning

According to I. Watson CBR is defined as *"Method of intelligent system that allows information managers to enhance the efficiency and to overcome the cost through automation of processes such as diagnosing, scheduling and designing. CBR functions by comparing the new problem with cases that are already achieved and then adapt the optimal solution according to current situation*" [13].

It is problem solving/analytical methodology which is basically dissimilar as compare to other paradigms provided by AI from various aspects. Rather than depending only on general knowledge regarding domain of problem or to develop any linkage with generalized interactions among problem description and result, CBR uses more precise, specific and definite knowledge related to past problem situations (cases) and their solution. A novel problem is solved on the basis of similar past problem (case) and then using it to resolve newly arrived similar problems. Moreover, CBR is a methodology which is characterized by continuous learning as every time a new experience is archived when problem is solved to provide solution for prospective problems [14].

## 1.7 CBR Life Cycle

CBR life cycle [15] has four processes as shown in figure 3:

### 1.7.1 Retrieve

In this process novel problem is solved by retrieving the prior most alike/similar cases. Case retrieval is the process of finding, within a case base, those cases that are the closest to the current case. To effectively retrieve the case, there must be the criteria for selection that finds out how the case is examined to be suitable for retrieving. Retrieval is the main area of research in CBR **.**Most common techniques for retrieval are:

#### 1.7.1.1 Nearest Neighbor Retrieval

In this technique, case is chosen when the features of case have greater weighted sum than all other cases in the case base. The features in case that are considered highly important are represented by high weights and less important features with less weight.

#### 1.7.1.2 Retrieval Knowledge Guided Approaches

They use knowledge to figure out those features that are important to retrieve the case in future. In some circumstances, various features of a case have various levels of significance to the level of success related to that case. These approaches may end result in a hierarchical structure that can be more valuable for case search.

#### 1.7.1.3 Inductive Approaches

They determine the significance of case features for discerning between identical cases, in result the hierarchical structure of the case base provides a compact searching for retrieving the case. In return this reduces the searching time of query.

#### 1.7.1.4 Validated Retrieval

It includes two phases. Phase one includes the retrieval of all those cases that seem related to a problem on the basis of major features of that case. Phase two involves figuring out more discerning facets from the preliminary group of those cases that are derived to find out whether these cases are suitable in the recent situation. Validated retrieval has benefit, as economical methods of computation can be utilized for the preliminary retrieval from the case base, whilst more costly methods of computation can be considered in the second phase.

### 1.7.2 Reuse

Reuse the previous cases that are retrieved.

### 1.7.3 Revise

Revise or adapt the case to solve the new problem. Adapting the case is the process of altering a solution that is retrieved into a solution which is suitable for existing problem. Adaptation is the most central step of CBR because it appends intelligence. A numerous approaches can be considered for case adaptation:

- The case retrieved can be used as a solution to the recent problem without alteration, or with alteration when solution is not suitable for the recent situation.
- The processes that were used to get the prior solution can be used without alterations or with modifications where the steps taken in the past solution are not totally acceptable in the up to date situation.
- A solution can be derived from numerous cases or, on the other hand, several alternative solutions can be offered if more than one case has been retrieved.

### 1.7.4 Retain

Retain the novel solution once it has been solved or confirmed.

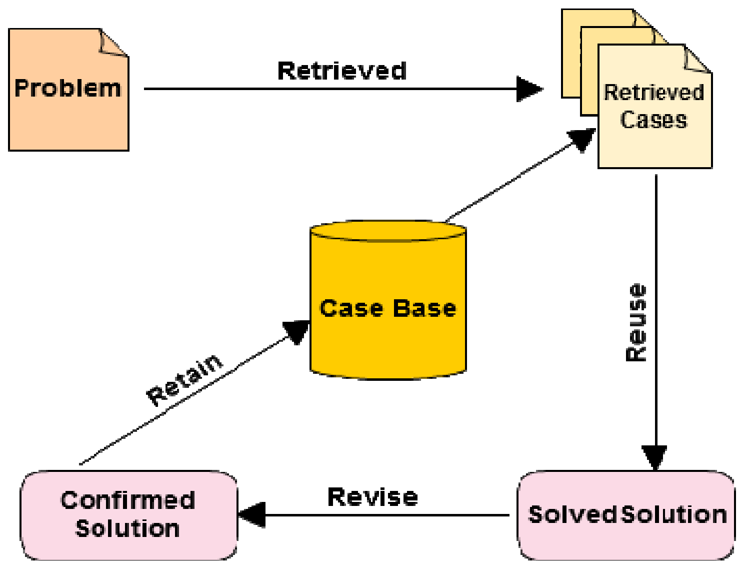


**Figure 3: CBR Life Cycle**

## 1.8 Machine learning and Case Based Reasoning

CBR is closely linked with machine learning in certain respects.CBR represents an approach of machine learning which is described by “lazy” learning and that is to store instance at the time of learning, deferring the inductive step till the problem is solved. According to Kibler and Aha: learning can be attained by keeping the exemplar that are showed to the learners and next when an unknown example is shown by allocating to it the conceptual class which belongs to best matched exemplar [15].Learning is inherited to CBR not only because that it encourages generalizations on the basis of similarity detection between cases but it also collects and indexes the cases in case base to use later. As a learning paradigm/pattern CBR has another advantage of storing the newly solved cases, rather than deriving the novel solution from start a CBR memorizes and adapts the old cases [16].

## 1.9 Problem Statement

By increasing the availability of similar web services, it has become a major problem to identify the best service from all available options. Service consumer must know about functionality of service as well as how well it can perform [17]. Furthermore, many facets need to be considered as web service selected by one service consumer may not be good for another [18].

A number of approaches have been proposed to handle web service selection problem but there are still some problems in these as:

- They require much effort and complex computations in selecting the best service.
- Current automated approaches for WS selection use functional or non-functional features of a service but do not incorporate Quality of Experience (QoE) (runtime behavior) for selecting web services during evaluation in response to a user query. QoE was introduced by Moorsel et al but no implementation detail was provided [19].
- They lack self-adaptation mechanism to avoid the failure in future.

## 1.10 Contribution

Our research work presents an intelligent framework for WS selection that has following contributions to research:

- An intelligent framework for WS selection based on CBR is proposed which provides optimal services to requesters.
- To incorporated Quality of Experience (QoE) (runtime behavior) in form of user reviews.

- To provide self adaption mechanism to avoid the failure in future. This is not provided currently. Our proposed framework uses CBR for learning/adaptation, although in [20] used CBR but only with functional features.
- To providing User control over search in form of preferences.

## 2. Dissertation Organization

The thesis is organized into chapters and thesis structure is shown in figure 4. Chapter 1 presents the brief introduction of terms used in thesis. Chapter 2 provides the literature review related to web service selection. Chapter 3 discusses the framework design in detail. Chapter 4 provides insight of implementation details and user interface of our proposed framework. Chapter 5 gives the analysis of results that are gathered from case based reasoning system. Chapter 6 concludes the thesis with further future work.

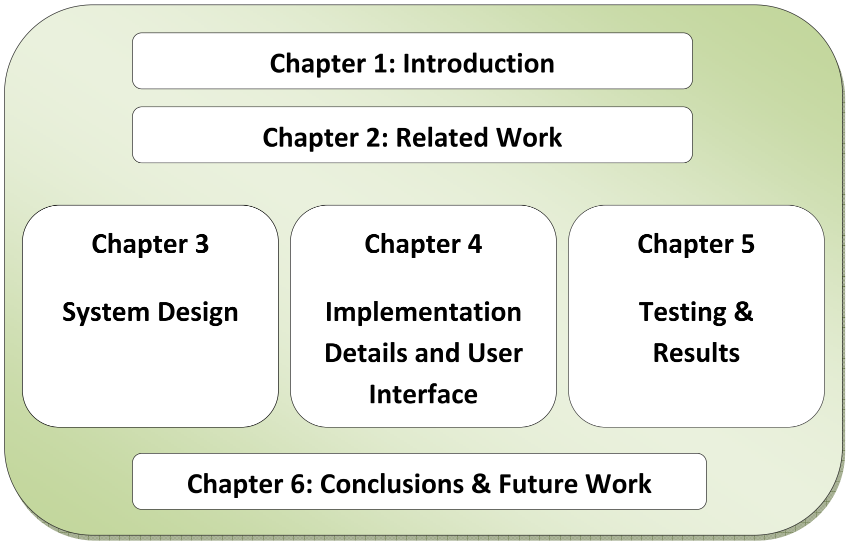


**Figure 4: Thesis Organization**

Chapter 2

# 2 Related Work

In [21] DQoS (Decision QoS) which is decision model for QoS has developed in order to evaluate the web services which are composed of constraints, decision models and extensible QoS models to enable the selection mechanism of web services on the basis of quality because it is crucial to consider the users' preferences and QoS to make decision effectively among listing of similar web services. Their proposed decision model consists of QoS criteria including cost of execution, reliability, execution time and availability to find web service, Decision modes to find which service to select on the basis of non-commensurate and contradictory Qos criteria. They defined four weight modes as subjective weight mode, objective weight mode, single weight mode and subjective-objective weight mode. Their DQoS model helps to choose services based on user defined preferences and it provides a solid theoretical basis to further work on dynamical composition of web service.

In [22] Umardand Shripad Manikrao et al have proposed framework which uses DAML document that describes functional, non functional requirements and contracts. This framework is basically for dynamically selecting the web service. Their framework deals with the inadequacy of orthodox web services by enhancing UDDI and WSDL by semantically describing the services and provides recommendation system which recommends the service from list of similar services. Service user provides his/her service requirements in terms of semantic document and service providers needs to enroll their services by using service descriptions which hold the profile of semantic service and QoS parameters as average response time, maximum time taken in execution and average time of execution etc. Semantic matcher uses semantic matching algorithm that matches the service request sent by user with services that are registered and then presents a service list matched with service request. This list is then passed to recommendation system which orders this list on the basis of knowledge learned from users' feedback. User then chooses service from this list. After the service is being provided to user, user provides rating for this service and this rating will show satisfaction of user for that service. Finally this rating is recorded in repository as it is used as input for recommendation system for future requests. This framework overcomes the limitations of orthodox web services by enhancing the capabilities of WSDL and UDDI and providing recommendation system to help in selection of web services from number of available similar services. Proposed framework is shown in figure 5.

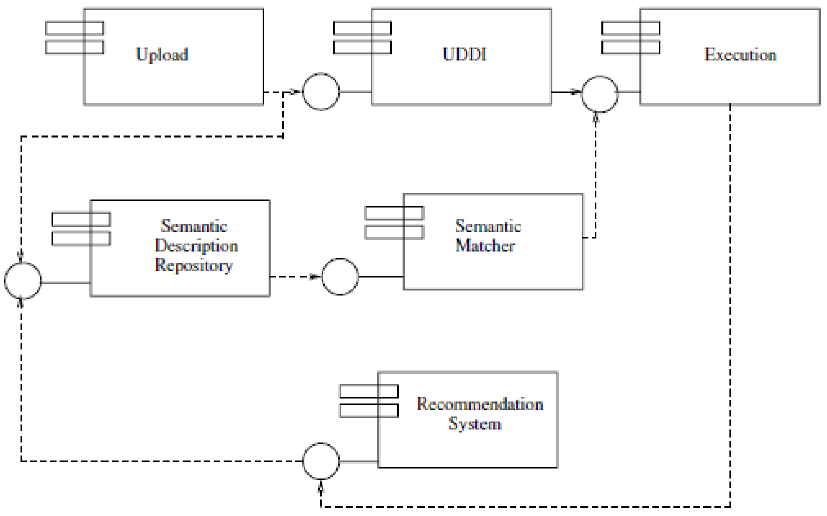


**Figure 5: Dynamic Web Service Selection Framework**

For QoS attribute a new terminology named verity is brought in by S. Kalepu et. al [23] and they developed architecture for its quantification. Verity is described as the measure of reliability and honesty of service providers. It is computed by the exterior components and is defined as the capability to keep lowest differentiation between the estimated and attained service metrics' level. They identified reputation as the vector of verity or service ranking .Their proposed architecture consists of service provider, broker, interceptor and users. Provider of Service publishes Service Level Agreement (SLA) enabled web service and then sends it to broker to store up in its repository. Service user registers with broker, it looks for and finalizes the SLA with suitable provider. Interceptor is liable for computing the SLA parameters, delivered by service invocation. Interceptor is initiated by user at the start of each service and when it is completed, interceptor passes on these values to user and broker for calculation verity.

They enhanced the functionality of user and the service broker by adding the verity calculator on both ends. They equipped databases (local and global compliance database) both at the user and broker ends for storing the records of all service invocations of past. Local database is contained with fewer records than global because local database contains only the records related to end user alone whilst global database contains the record of all the transactions of service broker.

Moreover, the user provides ranking for service and its provider at the end of each service completion. Local and global calculators take the aggregate of this ranking along with earlier rankings in order to get the average ranking. According to author only the user opinion is not enough to specify the reputation, it is essential to compute the trustworthiness of service provider to ensure the compliance with agreed/ contracted levels in SLA. Therefore to calculate the uniformity in compliance levels they introduced a new attribute in QoS termed as verity and to quantify it proposed architecture.

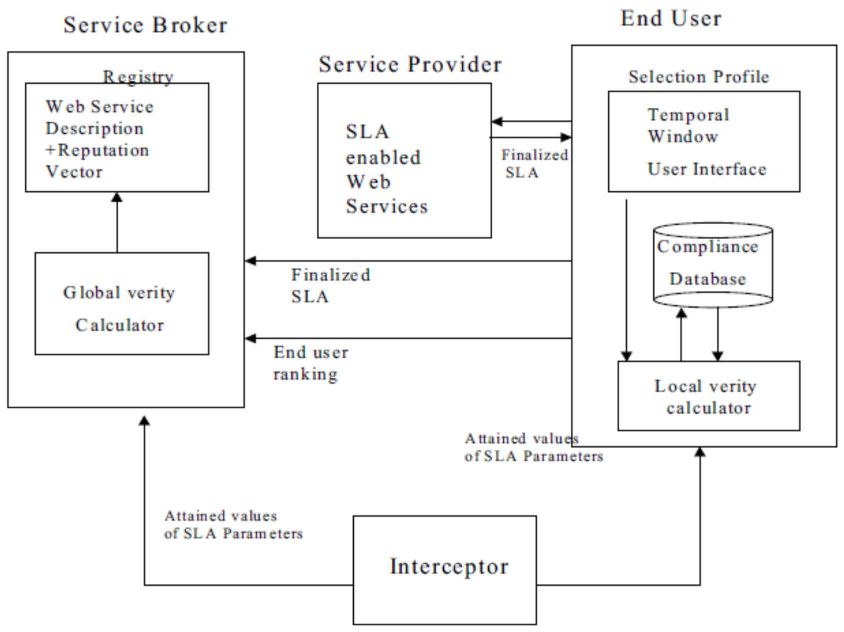


**Figure 6: Architecture for verity and reputation in SLA enabled Web services**

Miltiadis Lytras[24] have introduced a novel approach that assists the organizational knowledge flow by using the QoS . Author has introduced the concept of mediator for selection of web service on the basis of Web Service Resource Framework (WSRF). Apart from fulfilling the functional requirements, past domain knowledge is used by mediator for decision making about the ontology knowledge of QoS. Author's has developed WSRF specifications to help and explain the conventions of web services that make possible the interaction with resources in efficient, interoperable and standard way and then introduced architecture for web services by utilizing WSRF and QoS Broker between service provider and service consumer. This QoS Broker is autonomous which uses metadata of QoS related to providers that provide web services to consumers and also perform service selection decisions on the basis of quality. Broker performs as mediator between consumer and provider .It handles the set of web services and probes the service registry to find the availability of web services. When the web services are discovered /located successfully then their information, Uniform Resource Identifier (URI), function’s name and description is stored in database of QoS broker on the basis of categories. User sends service request to broker which will choose most appropriate service provider and then collects the QoS information for that provider and on arrival of the user request it decides which provider can satisfy not only the operational needs but QoS based also. If more than one services provided by service providers are satisfying the service consumer's need then decision algorithm decides which service to be selected that can better serve the user's request. Furthermore, feedback from users is also collected that helps broker to perform better decision making job. Their approach is novel as they have introduced a dynamic service knowledge management mediator which is based on WSRF, by this they introduced first time dynamic knowledge management for non functional requirements of web services.

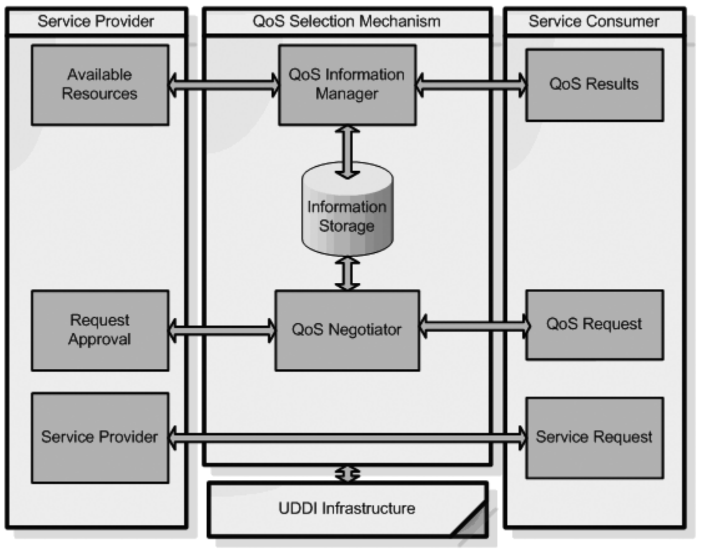


**Figure 7: Overview of Selection Mechanism**

E. Michael Maximilien et.al [25] have built a multi agent framework that is founded on QoS ontology and trust model. A foundation is provided by this ontology to help service providers in order to advertise their services and service consumers to convey their preferences and the service ratings to be collected and shared. These service ratings are necessary because they provide experimental basis for the choice of services. An ecosystem is formed by the agents in whom they assist each other. They practically evaluated their system through simulation.

According to [25] selection of services is performed on experimental basis by getting not only how the service is behaving and but also how it was being advertised. Users share their experience of using these services. A multi agent framework which makes use of a QoS ontology to help self-adjusting trust was built. The ontology assists the service providers in advertising the service offerings to service consumers so that they express their service preferences and ratings. These ratings play very important role as they establish the foundation for the trust in various implementations. They developed framework that extends the usual SOA with agents for the evaluation of trust model and estimation of self adjusting trust. They installed agents among each service and service consumer. These agents have same interface as the services have but they extend the interface of service with methods that are agent specific. By using the same interface like services these agents are capable of dynamically and clearly opt the service implementation by thinking the quality needs of consumers. Service consumer imparts its needs through the agent interface and service agents invoke the service methods and these agents monitor and pass on all service calls to that selected service. Rather than direct selection of implementation, service agent on the behalf of consumer selects the implementation which has more appropriate match with consumer's policy. Service agents also take part in agencies (nodes where agents share their opinions about quality) to share their view about service implementations that are selected by them. Quality concept is shared by agents in terms of ontology. Ontology is classified into three parts. The top QoS ontology has definitions of all qualities and relationship between them. The central QoS ontology augments the top ontology and explains the qualities that are appropriate for all domains. Bottom QoS ontologies depict the particular domains by augmenting the qualities in central ontology or generating the novel from the top ontologies. Their simulation results demonstrated that trust model generate a system that regulates the trust level for implementation of services by depending on the previous quality performance.

G. Vadivelou et.al[26] have introduced a novel architecture Delegation Web Service(DWS) as shown in figure 8, to choose the services efficiently and carry out load balancing. Alike web services have one DWS each and it also resolves load balancing issue.

For monitoring and observing the resources a standard is used called Web Service Distribution Management (WSDM). WSDM deals with management of web.

DWS hand over those requests which are functional to the subsequent web services but does not implement functional parts. It also does load balancing by getting all the web services that are requested and are similar together in register module and then allocate the priority to all web services in that group and uses the peculiar service having highest priority. If any service is overloaded then at once it switches to next service which comes after that in priority order.

Next DWS makes decision about the service which it should delegate. This decision is founded on requirements by service consumer which are functional and also non functional requirements in terms of QoS preferences. Firstly, consumer sends service request to DWS to attain best service by specifying functional requirements with QoS based prefereneces.DWS then looks in UDDI to find similar service in accordance with functional requirements and also sends the request to all web services to publish their QoS parameters. After sending the QoS parameters to DWS by services, DWS sends request to WSDM in order to evaluate the QoS parameters for service consumer’s preferences against the services with the help of their proposed service selection algorithm. In return, WSDM replies by presenting QoS metrics and suggests the best service. Then that peculiar best service is invoked and that responds to DWS and then finally that service is offered to service consumer. In their work load balancing facet is highlighted strongly. Therefore, each service is accompanied with threshold value and when that value is reached then it is overloaded. As DWS is endowed with each service therefore, to balance the load is easy because as there is overloading then the very next in already evaluated prioritized order is made available to consumer. In [26] QoS evaluation is made efficiently with WSDM and the best service is opted on the basis of proposed algorithm for service selection and as current web service selection approaches neither consider the issue of QoS efficiently nor load balancing is carried out to the maximum level.

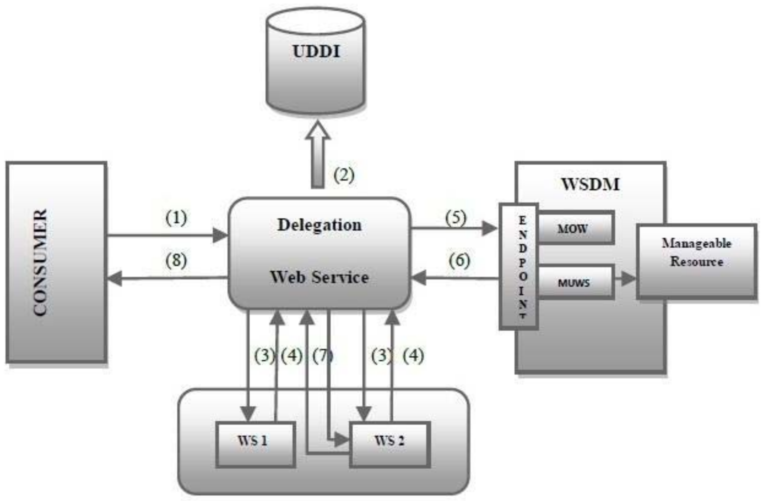
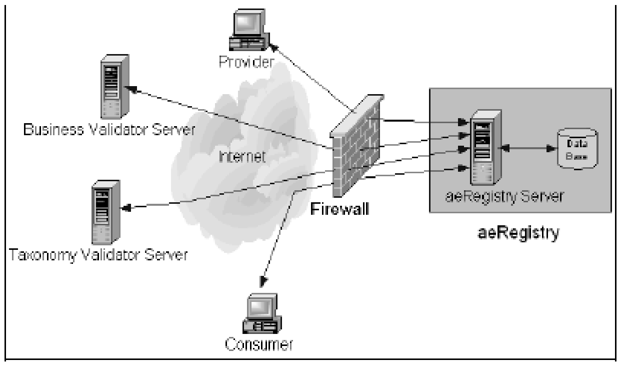


**Figure 8: Delegation Web Service Selection Architecture**

L.Taher et.al [27] have developed a framework called QoS Information and Computation (QoS_IC) framework as shown in figure 9 and 10 for web services on the basis of QoS selection mechanism. Their framework is composed of two models: Data Model and the other is Computation Model. Former consists of QoS Mode and registry ontology which not only gives semantic and properties of QoS but also the know-how of framework, whereas later deals with management of QoS and algorithm for QoS matchmaking. It also utilizes a technique of similarity distance measure in mechanism of QoS selection and gives notion of such a mechanism which is used to manage the dynamical alterations in QoS properties. Three main functionalities are being provided by this framework. Firstly, a mechanism for service selection based on QoS. A similarity distance Measure is used by this mechanism in the matchmaking process. In this matchmaking process the QoS based requirements specified by consumers are matched with providers’ published QoS information so that to find out the best match. Secondly, in case of dynamical changes in QoS characteristics/properties and thirdly creates the notification for service consumers if any changes in QoS properties are encountered. Moreover, framework is apprehended in aeRegistry which is infact the implementation of general Web service registry (UDDI).Their aeRegistry includes four  components: Update Manager which handles the request received from service providers in order to update registry in case of changes in QoS characteristics of published services and it also makes updations in QoS Manager along this it sends notifications to service consumers about QoS changes, Service consumer provides publish/search request which is received by SOAP request filter which filters these requests.

It uses SOAP header to discriminate between requests of consumer and provider. It is liable to transfer QoS parameters to QoS Manager. QoS Manager that performs two functions: updation of QoS characteristics of those services which are published as the data is obtained from update manager and it also provides selection algorithm which is based on QoS and Validation Manager that is liable to perform validation of taxonomy and information related to business supplied by providers during the registration process in registry. In this way, they built advanced framework for service selection based on QoS which retains standard UDDI architecture and added four more components init rather than making any alterations in current service registry in order to maintain the service quality and to offer a mechanism to update QoS. Moreover their approach is different as compare to others as no other approaches send notifications about variations in QoS properties to their consumers.



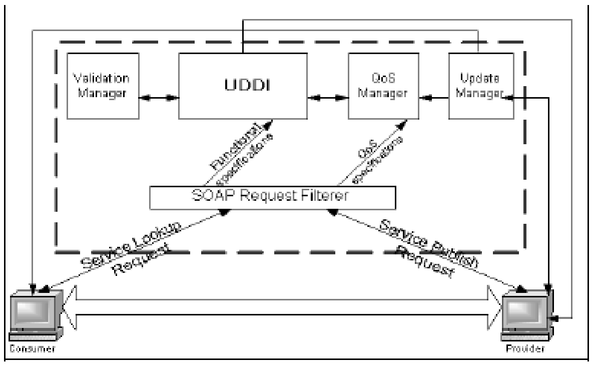


**Figure 9 : The Architecture of aeRegistry**

**Figure 10: QoS Information and Computation Framework**

Mick Kerrigan et.al [28] have presented an approach of web service selection in Web Service Execution Environment (WSMX) which is use by business communities and individuals. WSMX is implementation of Web Service Modeling Ontology (WSMO).It is ontology to describe the semantic of web services. It is founded on Web Service Modeling Framework (WSMF) as on four elements of it including Mediator, ontology, web services and goals. The purpose of WSMX is to create such environment to find and utilize the semantic web services in order to meet the users' need. WSMX facilitates user by presenting an automatic way of finding and using the services .Their WSMX approach consists of discovery, negotiation, selection, mediation and invocation steps. During discover process service is found that can fulfill the users' goals. After discovering the web service in negotiation step it is assured that discovered services can meet the user needs or not then those services which cannot meet the needs are eliminated from the discovered services' list. Next step is to select the service, after getting the list of services that can fulfill the user needs, now need to choose one of them according to users' preferences and WSMX has two types of preferences filtering and ordering. In filtering all the discovered web services are filtered and in ordering those filtered services are brought in order.

Both these preferences are provided by the service consumer. They have discussed both the automated and manual selection mechanisms. As manual web service selection is valuable for users but from business perception it takes the whole control out from providers' hand. Therefore, they have also described manual selection by step in between discover and invocation to allow business companies to contact with more than one service providers. They suggested that by combining the both approaches we can get the best out of them.

Abhishek Pandey et.al [29] have proposed a model as shown in figure11 for web service selection dynamically  by involving the clients in the web service selection process but the complexity of system is kept concealed from clients for security purpose. They have built their proposed model with service repository which behaves independently and forwards the requests of clients and security is also maintained by not allowing them to invoke it directly. This mechanism will help to forbid unauthorized clients to get access. This repository performs storing, reasoning and collection operations. During storing a feedback report of QoS is created by the client and stored in repository. This report acts as reference for service consumer to evaluate the service provider. Every service provider then maintains feedback information related to it. In collecting step all the essential data of service provider for the next step (reasoning operation) is retrieved. In reasoning step determines the best provider for the service to consumer is selected in accordance with the selected data. Unlike other approaches, this approach involves consumers in service selection process and their proposed model is based on algorithm for service selection which on the fly discards the service that is not matching and also provides transparent selection mechanism and assures the security but preventing unauthorized users.

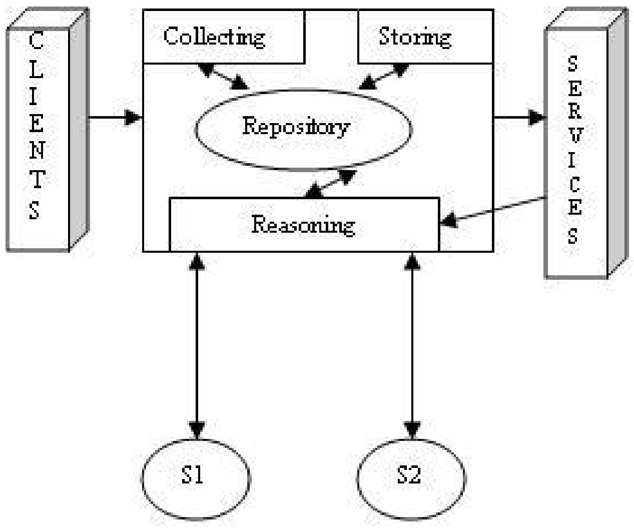


**Figure 11: Repository based Web Service Selection**

Olivia Graciela Fragoso Diaz et.al [20] have proposed a model for WS search and selection by using CBR as shown in figure 12 which permits the service providers that on the basis of functional requirements they describe their WS that allows service providers to describe their web services based on their functional characteristics. Their model is composed of UDDI, WSDL and library consists of cases and it categorizes the WS on the basis of functionality. Reasoner module uses two algorithms. One algorithm is used to find the required case in library and second algorithm is used for matching the user requirements with cases. First of all service consumers supply their requirements by using web interface, this interface sends these requirements to the reasoner module which selects various relevant service cases by using searching and matching algorithms. When WS is registered in UDDI, then reasoner module receives a unique identifier. This identifier helps to determine the category where the new service case depicted by WS can be indexed. Reasoner module also provides the set of identifiers to connection module which queries UDDI and in return the results are shown through web interface. Their model has inadequate mechanism in order to search and publish the WS but can be extended by using search engine.

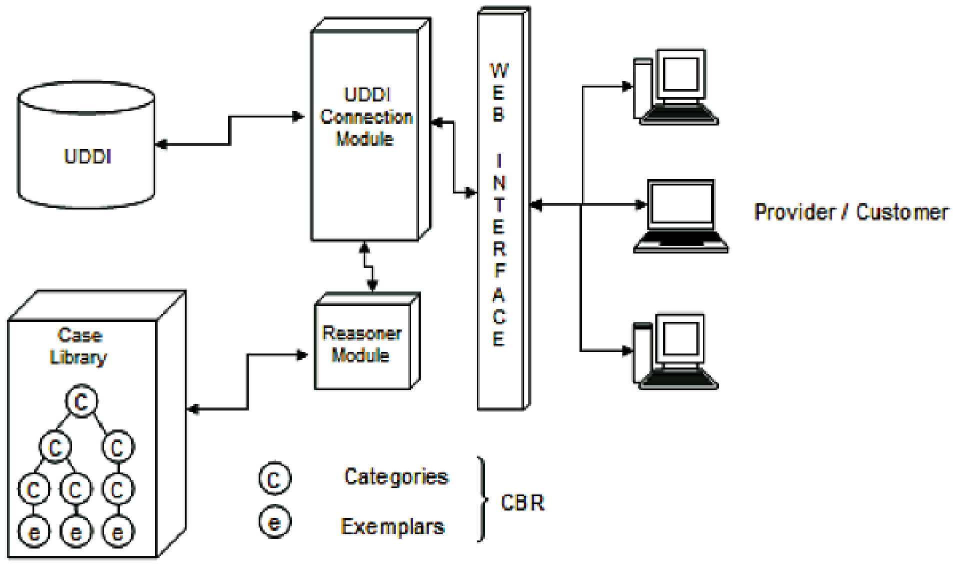


**Figure 12: Proposed Model Overview**

**Chapter 3**

# 3 System Design

In this chapter we shall discuss the details about our proposed framework. Section 3.1 discusses the proposed framework and its components. Working of our proposed framework is discussed in section 3.2.Section 3.3 gives insight about the testing and training of our framework and proposed algorithm is discussed in section 3.4.Section 3.5 describes an illustrative example by applying our proposed framework.

## 3.1 Proposed Framework

We have proposed "**Intelligent Web Service Selection Framework" (IWSSF)** based on CBR. CBR system remembers the previous situation (previous requests of customers) similar to the current one and uses that to fulfill new request. Our proposed framework is shown in figure 13.

There are three components in our proposed framework:

- Service Consumer
- Service Provider
- Case Based Reasoning System

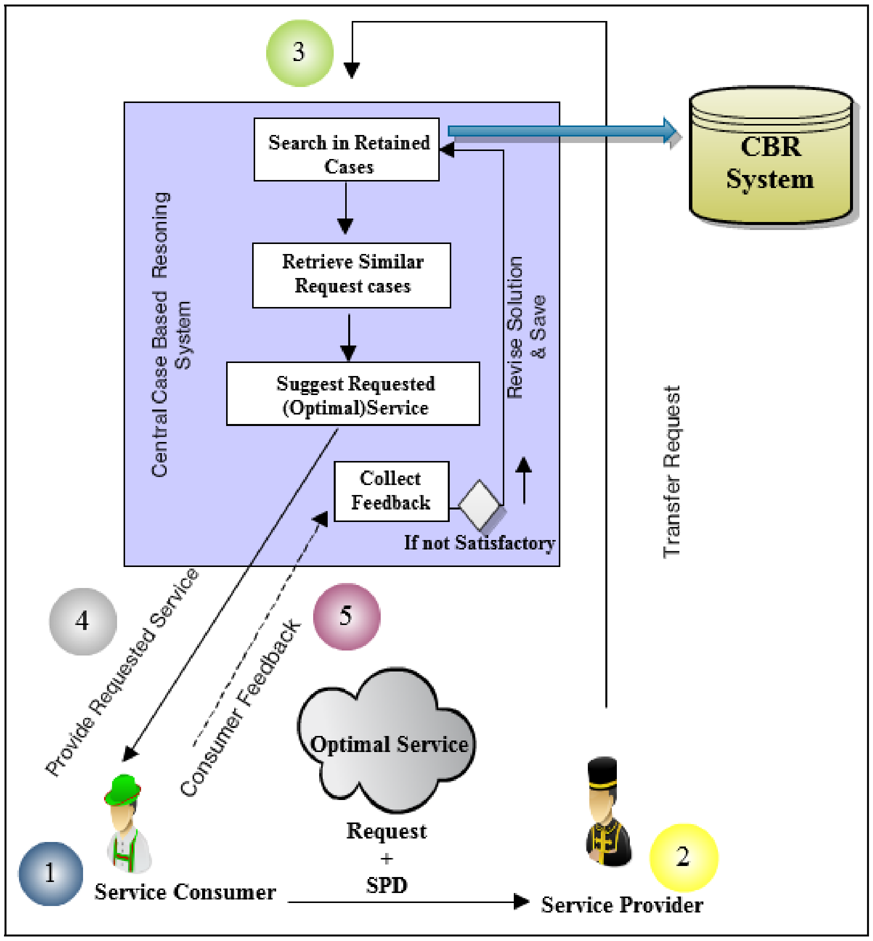


**Figure 13: Intelligent Web Service Selection Framework**

### 3.1.1 Service Consumer

It is software module /application/person or another service that calls for service. It searches for needed service in service registry, binds to that service and then executes the service operation.

### 3.1.2 Service Provider

It is any organization or person that publishes or registers its services in UDDI that is service registry where all the business bodies publish their services. It accepts the web service request of service consumer and executes these requests by providing their required services.

### 3.1.3 Case Based Reasoning System

The case base is a repository of experience, where each experience (called case) is generically of the form [input parameters, output solution]. Given a query, in the Retrieve step candidate solution are fetched from the case base on the basis of similarity between case parameters and query parameters. CBR uses our proposed algorithm to respond service consumers against their requests. Service provider will send service consumer's request to CBR for the optimal service selection .CBR then uses algorithm to compute the similarity of request for each service case that it retains and then responds to service consumer with those cases having highest similarity match with request and also collects feedback from consumer so that if any suggestions/improvements in service are needed then service could be revised, improved and in future only offer those service cases that satisfy the consumer s' needs well . Figure 14 shows the format of service cases in CBR system.

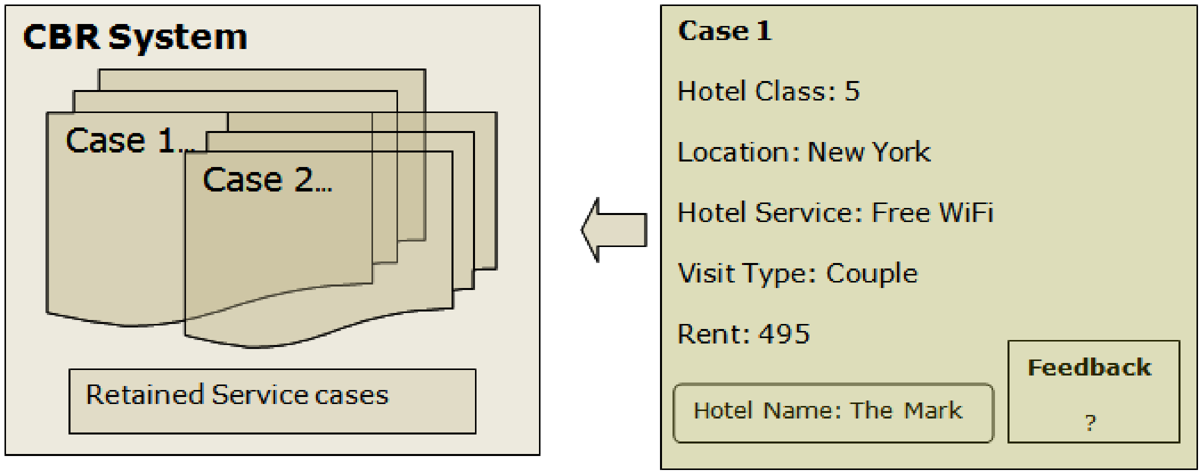


**Figure 14: Service Case Format**

## 3.2 Working of Proposed Framework:

Working of our proposed framework is shown in figure 15.Following steps define working of our proposed framework:

### 3.2.1 Input: Service Request

Input to the system is a service Request (SR) by the service consumer in the form of a SOAP message.SR contains service parameters along with their preference values as shown in equation 3.1; these preferences are specified in Service Preference Document (SPD) as shown in figure 14 which is sent along with SR; SPD contains preferences in terms of High, Medium and Low.

$$SR= [(p_1, f_1), (p_2, f_2), \dots,(p_n, f_n)] \dots\dots\dots\dots 3.1$$

where $p_i$, i=1,…,n are parameters and $f_i$, i=1,..,n are preferences.

### 3.2.1.1 Functional Parameters (FP)

These are tangible features that unequivocally describe functionality of a service from perspective of the service provider. These may have integer or continuous values. In our case, ‘price’ is functional parameter.

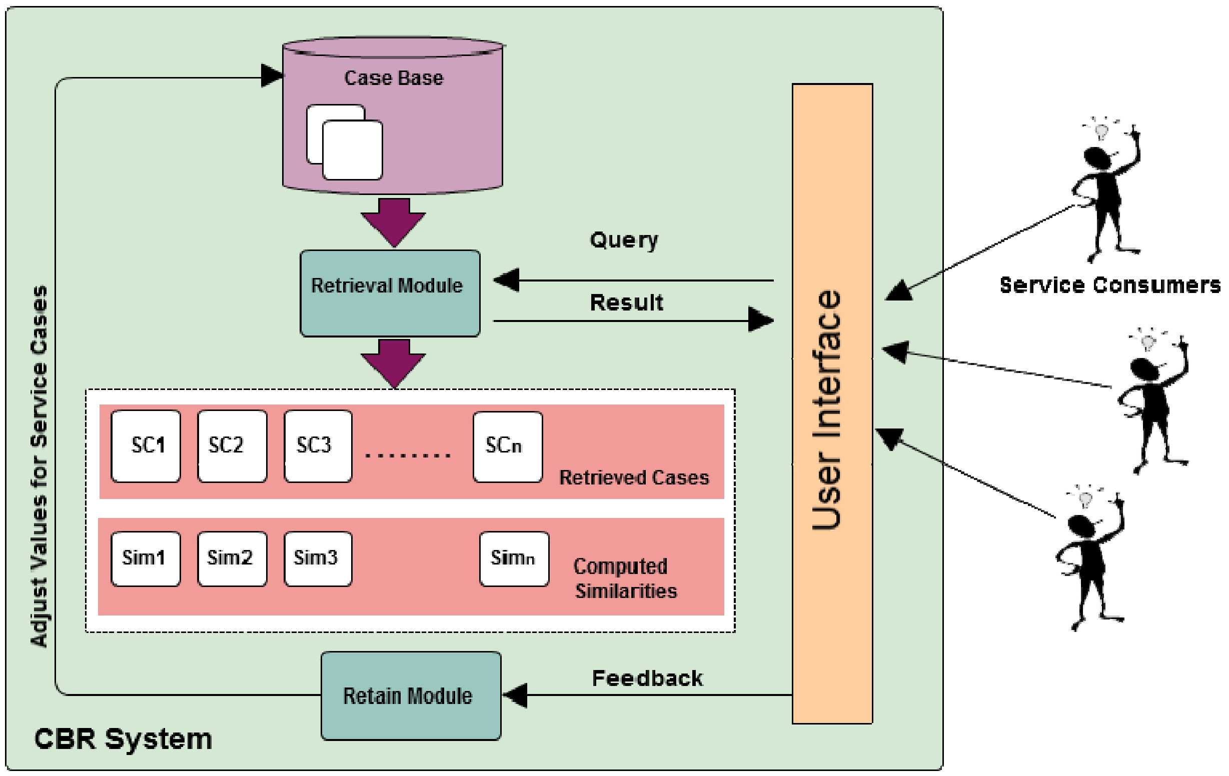


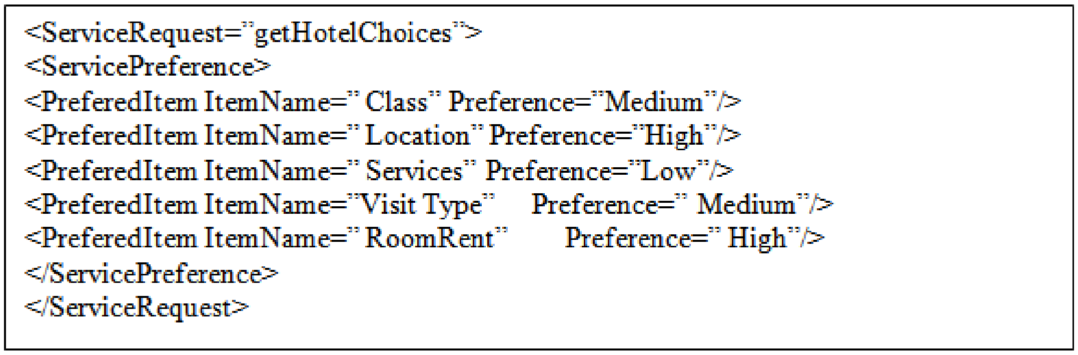
**Figure 15: Working of Intelligent Web Service Selection Framework**

### 3.2.1.2 QoE Parameters

QoE parameters represent the quality of users’ experience after using the service. In contrast to functional parameters, QoE parameters describe user’s perspective on the WS. Different users have different notions of quality. Moreover, users generally tend to describe the service in linguistic terms instead of formal definitions. In our case, ‘hotel service’ parameter (achieved from users' reviews) is QoE parameter. We recommend a domain-specific vocabulary for describing the QoE. This vocabulary can be populated offline by collecting linguistically described QoE of various users and finding frequently occurring expressions using text analysis techniques. Further, synonym thesaurus can be used to cluster similar meaning expressions in single class of QoE parameters. Data about 25 top hotels in New York was collected from TripAdvisor [30] which is the largest website for facilitating travelers to make trip plan for a perfect trip. QoE parameters are extracted from textual reviews by reviewers who had experienced the hotel by using Text Analyzer tool.

#### 3.2.1.3 Service Preference Document (SPD)

SPD includes preferences of the service consumers for the requested web service. SPD will be attached with each service request sent by the service consumer.



```
<ServiceRequest="getHotelChoices">
<ServicePreference>
<PreferedItem ItemName=" Class" Preference="Medium"/>
<PreferedItem ItemName=" Location" Preference="High"/>
<PreferedItem ItemName=" Services" Preference="Low"/>
<PreferedItem ItemName="Visit Type"    Preference=" Medium"/>
<PreferedItem ItemName=" RoomRent"     Preference=" High"/>
</ServicePreference>
</ServiceRequest>
```

**Figure 16: Service Preference Document**

On receiving the service request our proposed framework will retrieve all the request attributes and preferences and then after searching in the CBR system will return only relevant service cases that are closely matched to customer's request. On the other end CBR uses some weights as shown in Table 1 for these preferences whilst computing the similarity of each request against all the cases in CBR system.

**Table 1: Service Request Preferences**

| Preference | Weight |
|---|---|
| High | 3 |
| Medium | 2 |
| Low | 1 |

### 3.2.2 Search the Case Base

On receiving service request the discovery agent will look for required service in central Case Base. CBR system will retain various cases representing service queries, selected web services, and customers' feedback. Suppose there are currently n cases, $SC_1$, $SC_{2,}$… ,$SC_n$ in the case base, each identified by parameters *p*. Then each case can be represented as in equation 3.2.

$$SCi=[Input_i, Output_i, Feedback_i], \; i=1,2,\ldots,n \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. 3.2$$

Where

$Input_i=p_{i1}, p_{i2},\ldots,p_{ir}$

$Output_i$=Selected Web Service

$Feedback_i$=Feedback from the service user, measured on a scale of 3 (not satisfied) to 5 (highly satisfied).Initially it is set to 0 for all the service cases.

### 3.2.3 Similarity Measurement and Case Retrieval

Similarity computation is used to retrieve similar cases from the case base to answer the query. As the system allows both numeric and non-numeric parameters, different similarity measures are incorporated to handle different parameters. For numeric parameters, the system uses Canberra Similarity whereas for non-numeric parameters Cosine Similarity is used.

#### 3.2.3.1 Calculating Similarity at Parameter Level (Local Similarity)

To calculate the similarity of all the parameters of service consumer's request, our framework uses Cosine Similarity and Canberra Distance formulas depending up on the parameter type of service request as show in figure 17.

- **Similarity Computation of Numeric Values**

If the parameters in service request are of numeric type then proposed algorithm computes similarity by using Canberra Distance formula as shown in equation 3.3. This formula returns the difference between numeric parameters of service request and service case; finally by subtracting this difference from "1" we get the similarity.

$$d^{CAD}(SR,SC_i) = \sum_{i,j=0}^{n-1} \frac{|SC_i.p_{ij}.value - SR.p_j.value|}{|SC_i.p_{ij}.value + SR.p_j.value|} \ldots\ldots\ldots\ldots\ldots\ldots 3.3$$

$$sim_j(SR,SC_i) = 1 - d^{CAD}(SR,SC_i) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.4$$

Where SCi,pi and SR represents the service case, parameter and service request respectively.

When particular service request arrivers then for all the service cases in CBR system, parameters of numeric type are calculated by above formula by taking the absolute difference of parameter value of service case and service request and then dividing it with sum of their absolute value. Finally, this computed difference value is subtracted from one to get the similarity value as shown in equation 3.4.

- **Similarity Computation of Non-Numeric Values**

If the parameters in service request are of non-numeric type then proposed algorithm computes similarity by using Cosine Similarity formula as shown in equation 3.5.

$$(SR,SC_i) = \frac{TotalDotProduct}{\sqrt{\text{TotalSCCountSqr}}\sqrt{\text{TotalSRCountSqr}}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.5$$

$$DotProduct = (SCCount_k).(SRCount_h) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.6$$

$Count_k$ is number of times each character appears in both service case and service request.

1. For Cosine Similarity calculation first take the union of both parameters in service case and service request as shown in equation 3.7 (take similar character appearing more than one time once only).

$l1 = length(SC_i.p_{ij}.value)$

$l2 = length(SR.p_j.value)$

$l = l1 + l2$

$$UnionString = \{SC_i.p_{ij}.value.char_k | k = 1,2,\cdots,l1\} \cup \{\{ SR.p_j.val.char_h | h = 1,2,\cdots,l2\}\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.7$$

2. For every character in union, count how many times each of these characters appears in service case and service request(both service case and service request are treated as two vectors).

$SCCount_k = countCharAppearance(SC_i.p_{ij}.value.char_k, UnionString)$….3.8

$SRCount_h = countCharAppearance(SR.p_j.value.char_h, UnionString)$……3.9

3. Our interest is in $Count_k$ i.e. number of times each character appears in both service case and service request.
4. Take dot product of each character counted in service case and service request on the basis of its appearance as shown in equation 3.6.
5. Take square of each character counted in service case and add them then take square root of resulted values as shown in equation 4.1 and 4.2 respectively.

$SCCountSqr_k = Square(SCCount_k)$ ……………….4.1

Add $SCCountSqr_k$ to TotalSCCountSqr…………………4.2

6. Take square of each character counted in service request and add them then take square root of resulted values as shown in equation 4.3 and 4.4.

$SRCountSqr_h = Square(SRCount_h)$…………………..4.3

Add $SRCountSqr_h$ to TotalSRCountSqr………………....4.4

7. Multiply the resulted values obtained from step 5 and 6 to achieve the dot product as in equation 4.5.

$DotProduct = (SCCount_k).(SRCount_h)$……………4.5

8. Finally divide the dot product with resulted value achieved from step 7.

$$sim_j(SR,SC_i) = \frac{TotalDotProduct}{\sqrt{TotalSCCountSqr}\sqrt{TotalSRCountSqr}} \quad \ldots\ldots\ldots 4.6$$

### 3.2.3.2 Calculating Similarity at Service Case Level (Global Similarity)

Weighted Average is used to calculate the overall similarity of all service cases for each service consumer's request. Local similarities are now multiplied with preference weights. Customer specifies the preferences in terms of High, Medium and Low in SPD for each parameter in service request and weights are set for these preferences as discussed in section 3.4.3.

$$TotalSim(SR,SC_i) = \frac{\sum_{j=1}^{r} weight_j . sim_j(SR,SC_i)}{\sum_{j=1}^{r} weight_j} \ldots\ldots\ldots\ldots 4.5$$

Similarity $sim_j(SR,SC_i)$ of each parameter is multiplied with weight i.e. j=1 to r and then similarities of all parameters after multiplication with their corresponding weights are added; their sum is divided with total weight value which is sum of all the weights involve as shown in figure 17.

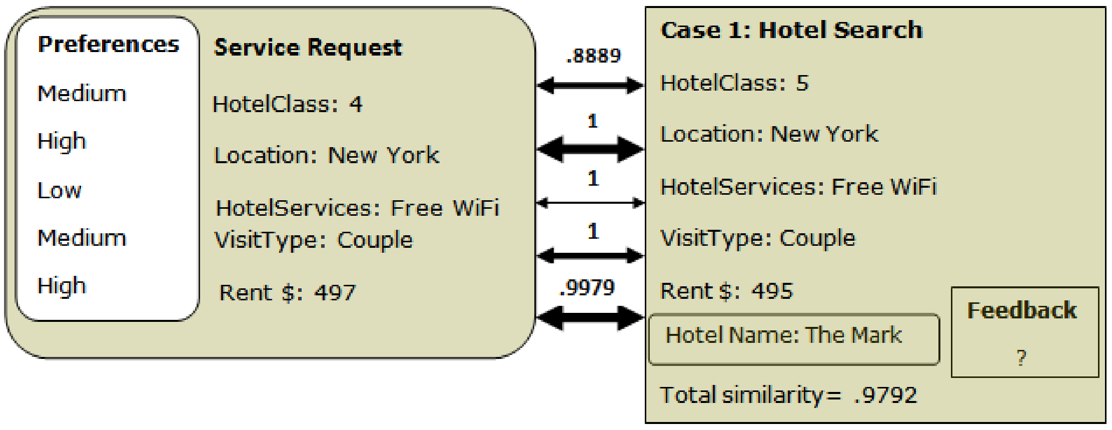


**Figure 17: Local and Global Similarity Computation**

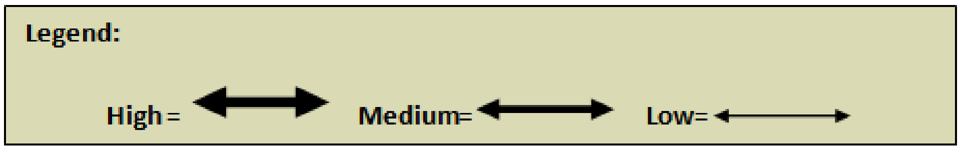


**Figure 18: Legend**

### 3.2.4 Feedback Collection and Case Revision

The main advantage of CBR based framework is that it can learn from experience and improve its performance in user satisfaction. Experiential knowledge is gathered in the form of user feedback once a service has been recommended and considered by the user. User's feedback can be satisfactory or not.

### 3.2.5 Learning on the Basis of Service Consumer's Feedback

On returning matched cases to the service requester our IWSSF also collects feedback from user. User's feedback is essential for CBR and user's feedback can be satisfactory or not. It is equally important for CBR to retain both, because it should know what the user really wants and to return satisfactory services. Our proposed framework's learning is based on user's feedback.CBR system returns cases on the basis of similarity computation and in case of same similarity of multiple service cases it looks feedback and then returns those cases having high feedback. User provides feedback in terms of:

- Highly Satisfied
- Neutral
- Not satisfied

We have set some weights for feedback .Weighting mechanism is shown in Table 2.

**Table 2: Feedback Weighting Mechanism**

| Feedback | Weight |
|---|---|
| High Satisfied | 5 |
| Neutral | 3 |
| Not Satisfied | 1 |

### 3.2.6 Calculating User's Feedback (Average Feedback)

Our IWSSF uses both similarity and average feedback to retrieve service cases upon service consumer's request and returns those cases having high similarity and feedback. It also computes average feedback. Let suppose more than one service consumers provide feedback for same case. CBR takes average of these feedbacks.

## 3.3 Training and Testing of Proposed Framework

Figure 19 shows the training of our proposed framework. Functional parameters are provided by service provider. Feedback and QoE parameters are provided by service consumer. Our Service case based system contains different service cases which are combination of these functional and QoE parameters. During testing as shown in figure 20, service consumer will provide functional and QoE parameters along with preferences and feedback (initially the feedback of all the service cases is zero). Functional and QoE parameters are sent to Retrieval Module and feedback is sent to Retain Module which adjusts the values for case entries as discussed in section 3.2.5 .Retrieval

Module passes parameters to Case Base where they are matched with other cases in Case Base and then return the list of cases on basis of similarity back to Retrieval Module which in return suggests the services to service consumer.

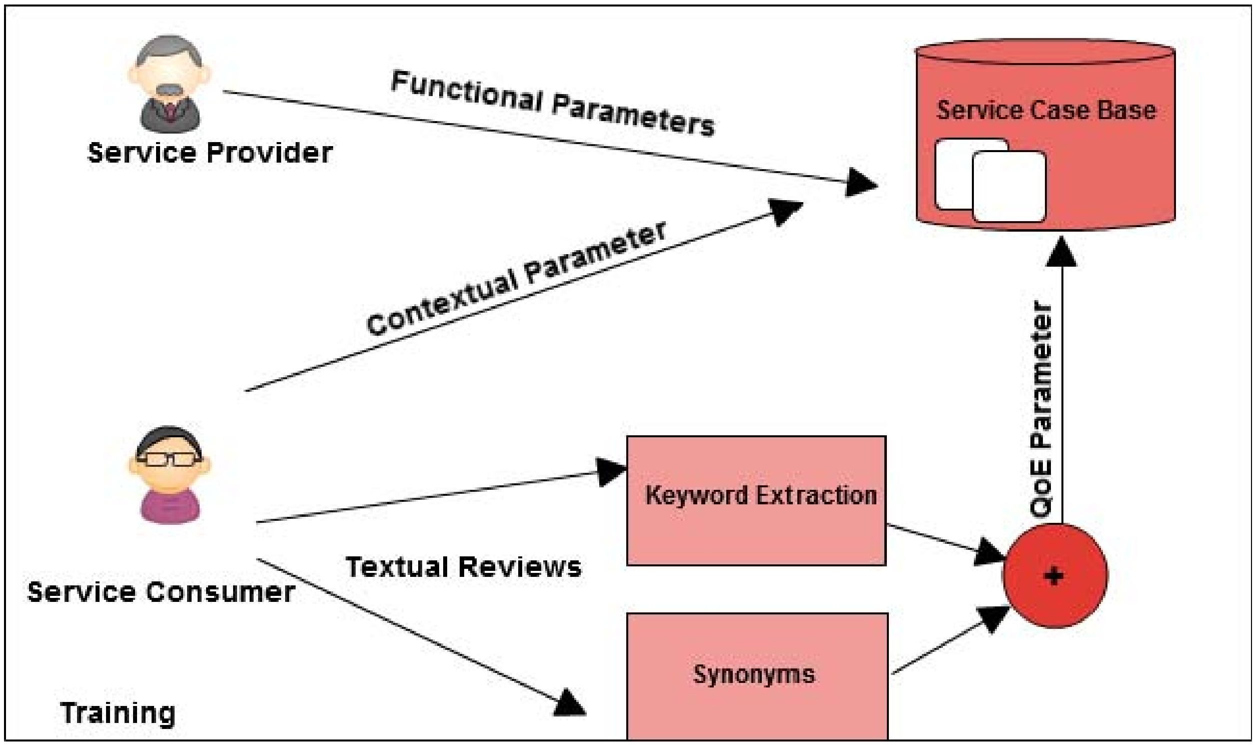


**Figure 19: Training of Proposed Framework**

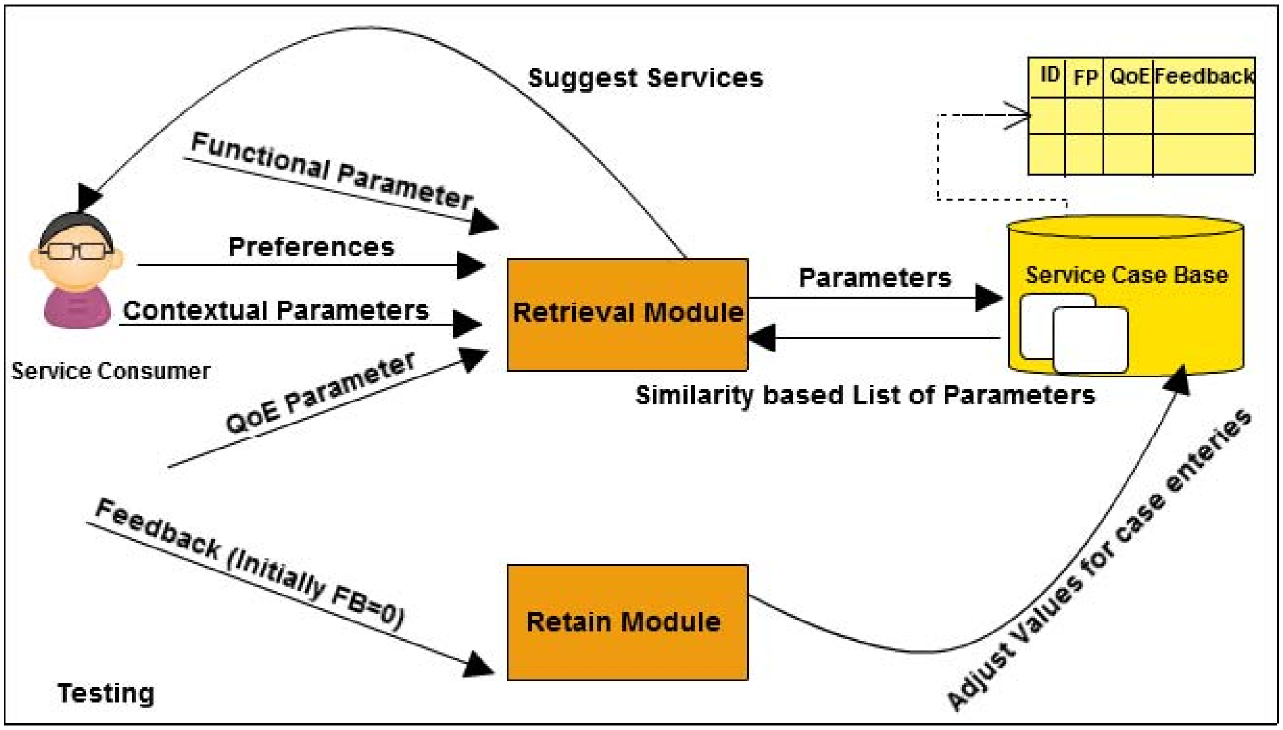


**Figure 20: Testing of Proposed Framework**

## 3.4 Service Request Matching Algorithm (SRMA)

```
SRMA (in: SR, $SC_1$, $SC_2$, …,$SC_{n;}$ out: R)

//input parameters:  SR is service request of format as in (3.1)
//$SC_1$, $SC_2$, …, SCn are cases stored in case base
//output parameter: R is a set of recommended services
begin
for  all cases SCi(i=1,..,n) in casebase
{
for all parameters $p_{ij}$ (j=1,…, r) in SCi
{
```

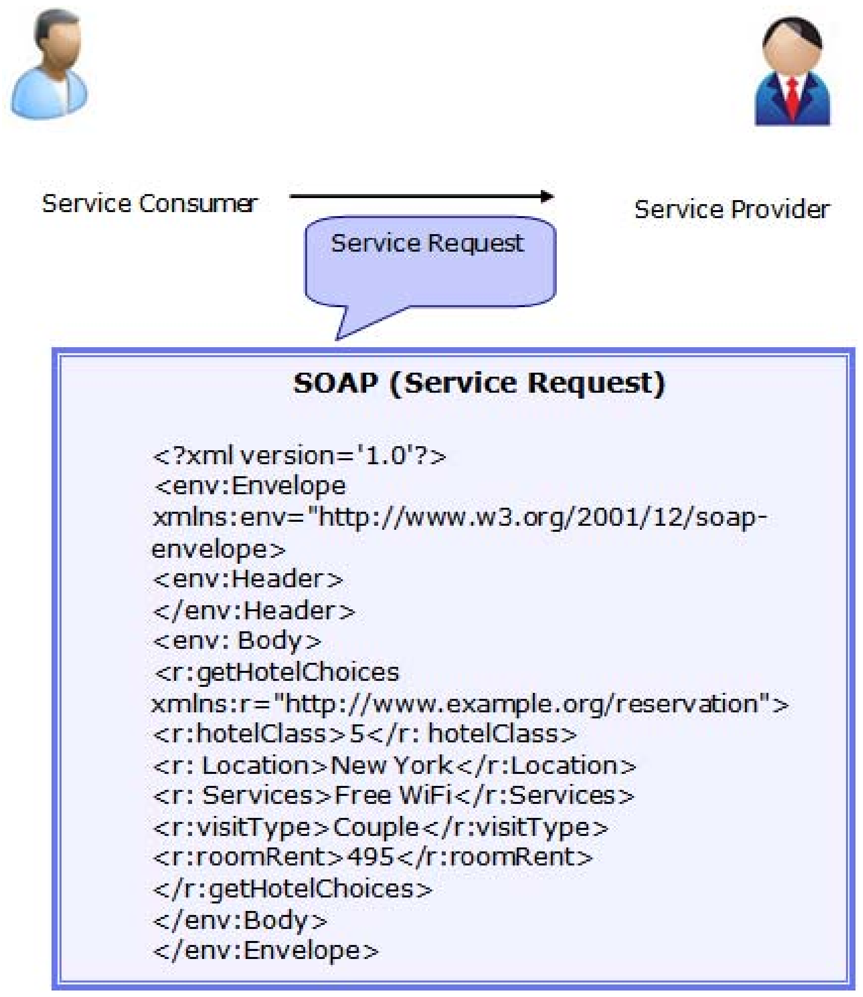
```
If $p_j$.type== "numeric" then
//find Canberra Similarity
$d^{CAD}(SR,SC_i) - \sum_{i,j=0}^{n-1} \frac{|SC_i.p_{ij}.value - SR.p_j.value|}{|SC_i.p_{ij}.value + SR.p_j.value|}$
$sim_j(SR,SC_i) = 1 - d^{CAD}(SR,SC_i)$
else
// use Cosine Similarity
$l1 = length(SC_i.p_{ij}.value)$
$l2 = length(SR.p_j.value)$
$l = l1 + l2$
$UnionString = \{SC_i.p_{ij}.value.char_k | k = 1,2,\cdots,l1\} \cup \{\{SR.p_j.value.char_h | h = 1,2,\cdots,l2\}\}$
// Count frequency of each character of //Union String in Service Case and //Service Request separately
for all $c_k$ in UnionString k=1,…, $l1$
$SCCount_k = countCharAppearance(SC_i.p_{ij}.value.char_k, UnionString)$
$SCCountSqr_k = Square(SCCount_k)$
Add $SCCountSqr_k$ to TotalSCCountSqr
$SRCount_k = countCharAppearance(SR.p_j.value.char_k, UnionString)$
$SRCountSqr_k = Square(SRCount_k)$
Add $SRCountSqr_k$ to TotalSRCountSqr
```

```
$DotProduct_k = (SCCount_k).(SRCount_k)$
Add $DotProduct_k$ to $TotalDotProduct$
end for
$sim_j(SR, SC_i) = \frac{TotalDotProduct}{\sqrt{\text{TotalSCCountSqr}}\sqrt{\text{TotalSRCountSqr}}}$
end else
end for
if SR.pj.priority == ”High” thenWeightj=3
else if SR.pj.priority == ”Medium” thenWeightj=2
else if SR.pj.priority== “Low” thenWeightj=1
//find weighted similarity, based on user //preferences
$TotalSim(SR, SC_i) = \frac{\sum_{j=1}^{r} weight_j . sim_j(SR, SC_i)}{\sum_{j=1}^{r} weight_j}$
end for
$Output_R = SC_i.Output \mid (TotalSim(SR, SC_i) > (TotalSim(SR, SC_s))OR$
$((TotalSim(SR, SC_i) == (TotalSim(SR, SC_s)\ AND\ (SC_i.Feedback > SC_s.Feedback))\forall s \neq$
$i, s = 1, \ldots, n$
end
```

## 3.5 Illustrative Example

We consider a scenario of hotel search in which service consumer wants to reserve a room in hotel. The following steps illustrate how our proposed framework

- After login Service consumer sends request "getHotelChoices" as shown in figure 21 , to search the hotels including parameters as 'HotelClass', 'Location', 'Visit Type', 'RoomRent' and 'Services' along with Service Preference Document (SPD) which contains preferences in terms of high ,medium or low against each parameter to service provider.
- This request will be sent as Simple Object Access Protocol (SOAP) message [30] which is a very light weight protocol used for the exchange of structured information in distributed and decentralized environment. It uses XML for its format [31].
- Service provider transfers this request to CBR system for matching.
- CBR looks request parameters and preferences, for these preferences it considers different weights as shown in Table 1.
- CBR matches the service request parameters with all the service cases that are stored in CBR.
- It computes the similarity of parameters by using our request matching algorithm and also considers the feedback.
- It returns those service cases to service provider whom similarity and average feedback is high and in our scenario it responds with those hotels which are more closely matched with request parameters after computing the similarity.
- Service provider responds to service consumer with highly matched service cases that meet the consumer's specified criteria.
- System keeps record of all the service requests made by service consumer so that he/she can record his/her feedback regarding service after experiencing those services



```
<?xml version='1.0'?>
<env:Envelope
xmlns:env="http://www.w3.org/2001/12/soap-
envelope>
<env:Header>
</env:Header>
<env: Body>
<r:getHotelChoices
xmlns:r="http://www.example.org/reservation">
<r:hotelClass>5</r: hotelClass>
<r: Location>New York</r:Location>
<r: Services>Free WiFi</r:Services>
<r:visitType>Couple</r:visitType>
<r:roomRent>495</r:roomRent>
</r:getHotelChoices>
</env:Body>
</env:Envelope>
```

**Figure 21: Service Request**

Chapter 4

# 4 Implementation Details and User Interface

We have developed a web based application for optimal selection of web service by using CBR. In this chapter we shall discuss the implementation details regarding the development of web application and user interface. Implementation details are discussed in section 4.1 and section 4.2 gives detail about web service creation in .NET framework. Section 4.3, 4.4 and 4.5 discusses the benefits of WS, life cycle of WS and the user interface respectively.

## 4.1 Implementation Details

We have developed our web service by using .NET development technologies. For this purpose we have used C# language (C#.NET) and Microsoft Visual Studio 2010 is used as Integrated Development Environment with the SQL Server as backend database.

Microsoft Visual Studio is used to create console applications, graphical user interface including web application and web service. There is a vast library which offers interoperability of languages. Therefore, it also supports various programmable languages which include VB.NET, C#.NET, J#.NET. It also helps to provide support of various other programming languages like Ruby and Python through language services which are installed separately. Support of HTML, CSS, XML, JavaScript, XHTML and XSLT is also provided [32].

## 4.2 Web Services in .Net Framework

WS is class which has methods and these can be invoked by methods which are on other computers with the help of protocols and data formats e.g. HTTP (Hyper Text Transfer Protocol) and XML. In .NET, SOAP is used to implement the method calls on network. Applications transfer and present data in XML-based format with the help of SOAP.

With the emergence of web services now organizations have moved towards web services. Therefore .NET framework assists in providing a very easy and user friendly method to make the web services[33].Web services are introduced as a part of architecture by .NET framework, it makes easy to develop and use these services by writing only few lines of code. Most internal logic to grip the remote method calls is abstracted by .NET framework. Visual studio .NET provides this support in development environment. Three major components as shown in figure 22 that make up a Web Service are:

- Web service on server
- Client side application that calls the web service using web reference
- WSDL that illustrates the function of the Web Service.

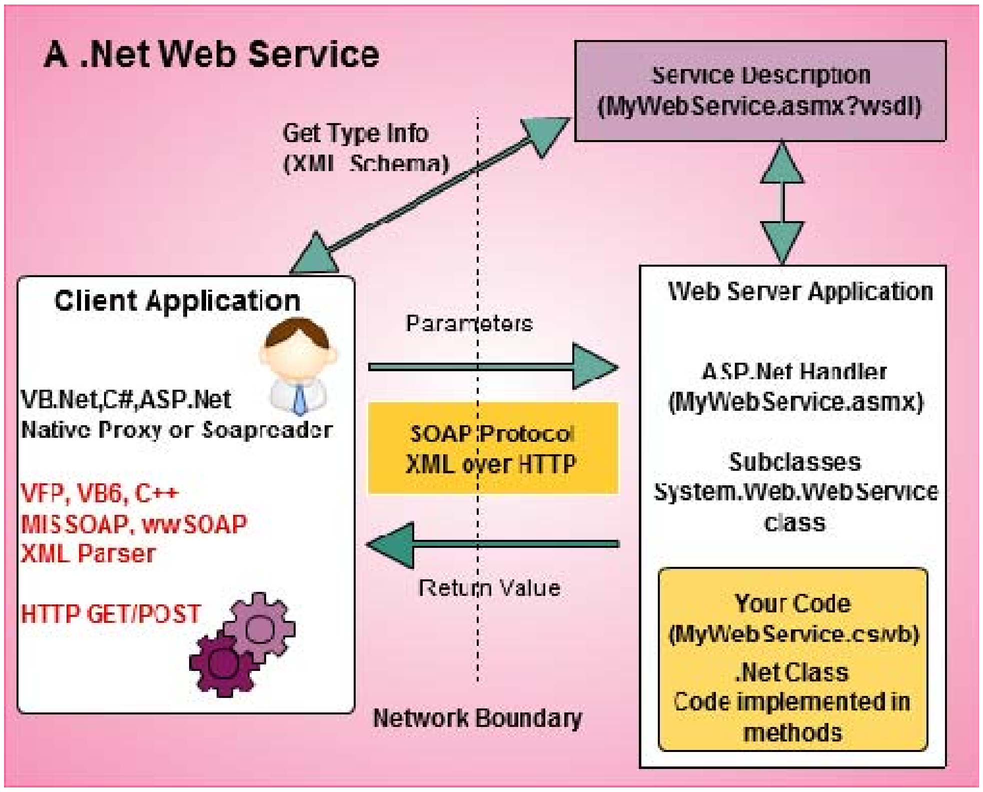

A .Net Web Service
Service Description (MyWebService.asmx?wsdl)
Get Type Info (XML Schema)
Client Application
VB.Net,C#,ASP.Net
Native Proxy or Soapreader
VFP, VB6, C++
MSSOAP, wwSOAP
XML Parser
HTTP GET/POST
Parameters
SOAP Protocol XML over HTTP
Return Value
Network Boundary
Web Server Application
ASP.Net Handler (MyWebService.asmx)
Subclasses System.Web.WebService class
Your Code (MyWebService.cs/vb)
.Net Class
Code implemented in methods

**Figure 22: Components of Web Service [33]**

In .NET WS contains .asmx page that has class which gives functionality of web service or the reference of particular external class that deals with the logic in class file which is external. When .asmx page is developed then the web service is ready to access on the web.

There are three methods to run the .NET web service.

**1. HTTP Get Operation**

In this way arguments are passed to WS by invoking ASMX page by providing the query string arguments to methods in order to call them.

Example: MyProject.asmx/MethodName? Parm1=value

**2. HTTP POST Operation**

It works in the same manner as GET operation but the arguments are passed in terms of Uniform Resource Locator (URL) encoded form.

Example:MyProject.asmx/MethodName

**3. SOAP**

In .NET  a proper way to call the  web servive is SOAP and it is used by  .NET internally for calling the web services [34].

## 4.3 Benefits of Web Services

Benefits of WS are as following:

### 4.3.1 Easy Accessibility

WS are easy to access as computer connected to internet can access the WS.web applications that are located at different computers using different platforms can easily exchange the data.

### 4.3.2 Flexibility in Structure

WS have flexible structure as various WS can be developed on the basis of requirements. For example we can create such WS that provides the basic functionality and can be used in various applications.

### 4.3.3 Easy Integration

Various busuness applications that are developed using different softwares and hardwares can easily be integrated in a very less cost.

## 4.4 Web Services' Life cycle

There are various steps involve in the development of web service as shown in figure 23. These various steps are:

- WS Development
- Publishing WS
- Locating WS
- Accessing a WS

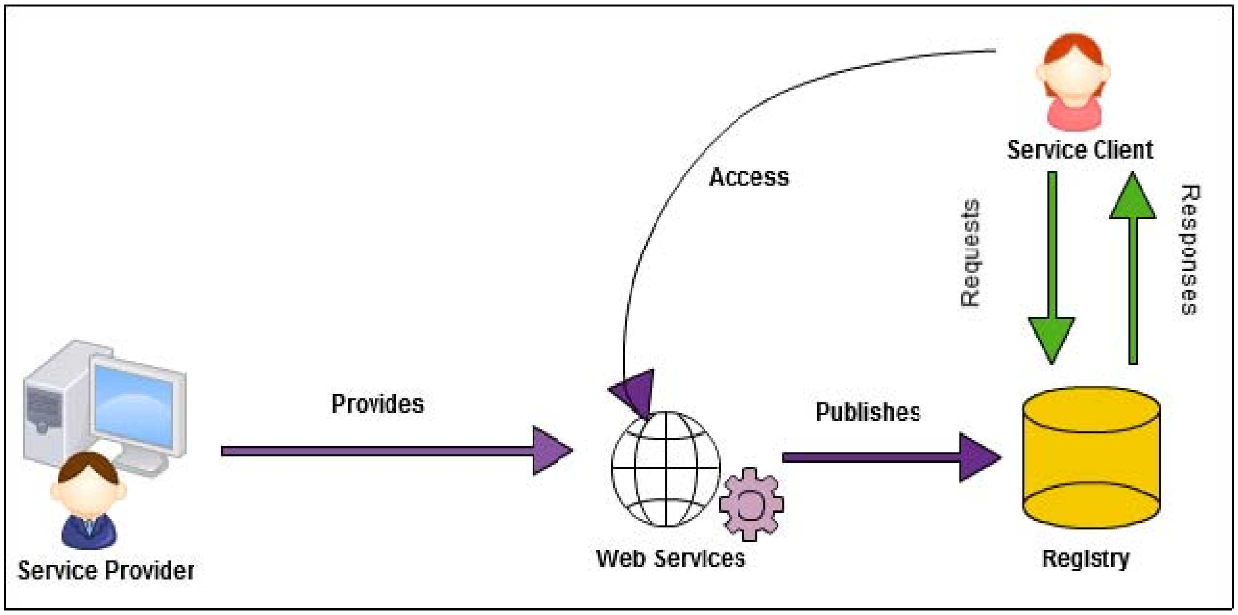


**Figure 23:Web Service Life Cycle**

### 4.4.1 Developing a Web Service

At this step it involves describing the WS interface and business functions are implemented. Interface contains various methods that are called by service users.

### 4.4.2 Publishing a Web Service

Service providers publish their services in service registry i.e. UDDI which keeps all the information about the WS and how to access that service.

### 4.4.3 Locating a Web Service

Service consumer searches for WS in service registry. When it is found then it gets the service information that is necessary to get the web service.

### 4.4.4 Accessing a Web Service

When WS is located then service consumer is connected with WS for accessing the services. It uses the URL to access the WS which is provided by the service provider [35].

## 4.5 Code Discussion

As stated earlier that we have developed our web service in C#.NET, In this section the code of main methods in our web service is discussed in detail. Our proposed SRM algorithm operates accordingly on the basis of parameters' type. As discussed in section 3.2.3.1 if parameters are numeric then Canberra Distance formula is being used by our algorithm for similarity computation. getSimilarityCanberra() method performs similarity computation by taking two parameters service case present in case base and request case and returns computed similarity of numeric parameters.

Below is C#.NET code of getSimilarityCanberra() method.

```
[WebMethod] // Method using Canberra Distance formula to compute numeric similarity

 public double getSimilarityCanberra(double ServiceCase, double ReqCase) {

 double Similarity = 1 - (Math.Abs(ServiceCase - ReqCase) / (Math.Abs(ServiceCase)
 +    Math.Abs(ReqCase)));
 return Similarity;    }
```

To calculate the similarity of non-numeric parameters our algorithm uses CosineSimilarity formula. getSimilarityCosine() method performs similarity computation by taking two parameters service case present in case base and request case and returns computed similarity of non-numeric parameters. Steps of this similarity computation method have been discussed in detail in section 3.2.3.1 of chapter 3 while C#.NET code of getSimilarityCosine() method is shown below:

```
[WebMethod] // Method using Cosine Similarity formula to compute non-numeric similarity
   public double getSimilarityCosine(string CaseStr, string ReqStr){
   CaseStr = TrimSymbolsDigits(CaseStr);
   ReqStr = TrimSymbolsDigits(ReqStr);
  double Similarity = 0;
  string UnionOfStrs = strUnion(CaseStr + ReqStr);
  string NoCharCase = CountCharAppearance(CaseStr, UnionOfStrs);
  string NoCharReq = CountCharAppearance(ReqStr, UnionOfStrs);

  double DotProdAB = dotProduct(NoCharCase, NoCharReq);

  try {

  Similarity = DotProdAB / (Math.Sqrt(SqrAddCmaSprStr(NoCharCase)) *

  Math.Sqrt(SqrAddCmaSprStr(NoCharReq))); }

  catch (Exception exptn) {

  Similarity = 0;    }

   return Similarity;
       }
```

In getSimilarityCosine() method which is shown above, TrimSymbolsDigits() method is called first to trim all the digits and symbols from non-numeric input parameters.C#.NET code for trimming symbols and digits from input string is shown below:

```
[WebMethod]  // Trim method to remove all symbols and digits from input

 public string TrimSymbolsDigits(string strToTrim){

 strToTrim = strToTrim.Trim();

 char[] ArrStr = strToTrim.ToCharArray();

 string Trimed = "";

 for (int i = 0; i < ArrStr.Length; i++)  {

 if (char.IsLetter(ArrStr[i]))

 Trimed = Trimed + ArrStr[i].ToString();  }

 return Trimed;                }
```

After trimming the second method called in getSimilarityCosine() method is strUnion() method which will take the union of non-numeric parameters of service case and request case as discussed in equation 3.7 in section 3.2.3.1.C#.NET code of strUnion() method is shown below:

```
   [WebMethod] //Method to take union of both service request and service case strings
 public string strUnion(string CharUnion){
CharUnion = CharUnion.Trim();
CharUnion = CharUnion.ToUpper();
CharUnion = CharUnion + "$"; // $ is end of String/Array
char[] SimChar = CharUnion.ToCharArray();
int i = 0, j = 0;
while (SimChar[i] != '$') {
 j = i + 1;
while (SimChar[j] != '$') {
if (SimChar[i] == SimChar[j]) {
int k = j;
while (SimChar[k] != '$')  {
SimChar[k] = SimChar[k + 1];
k++;      }   }
if (SimChar[i] == SimChar[j])
continue;
else
 j++;      }
i++;    }

CharUnion = "";

i = 0;

while (SimChar[i] != '$')   {

CharUnion = CharUnion + SimChar[i].ToString();

 i++;    }

 return CharUnion;    }
```

For Union string which is achieved by strUnion() method, our next method called in getSimilarityCosine () method is CountCharAppearance() method will count how many times each of these characters appears in service case and service request as discussed in depth in equation 3.8 and 3.9 in section 3.2.3.1 of chapter 3.

```
[WebMethod]

 public string CountCharAppearance(string CaseStr, string UnionStr) // returns the Comma

 separated numeric values        {

 char[] ACharCase = CaseStr.ToUpper().ToCharArray();

   UnionStr = UnionStr + "$";

  char[] ACharUnion = UnionStr.ToCharArray();

  UnionStr = "";

  int i = 0;

  while (ACharUnion[i] != '$')   {

  int CountChar = 0;
  for (int j = 0; j < ACharCase.Length; j++)   {


 if (ACharUnion[i] == ACharCase[j])

  CountChar++;      }

  i++;

  UnionStr = UnionStr + "," + CountChar.ToString();     }

  UnionStr = UnionStr.TrimStart(',');

   return UnionStr;     }
```

Next method called in getSimilarityCosine () method is dotProduct () method takes the dot product of each character counted in service case and service request by CountCharAppearance (). dotProduct () method is explained in detail in chapter 3.

```
[WebMethod]
 public double dotProduct(string Values1, string Values2){

 double Product = 0;

 string[] ValCh1 = Values1.Split(',');

 string[] ValCh2 = Values2.Split(',');

  try{

  for (int i = 0; i < ValCh1.Length; i++) {

  Product = Product + double.Parse(ValCh1[i]) * double.Parse(ValCh2[i]); }  }

  catch (Exception exptn) {

  Product = 0;    }

  return Product;   }
```

In getSimilarityCosine () method, last step is to take square of each character counted in service case and service request separately, add their squares individually and finally take square root of resultant values (achieved after adding squares) separately for service case and request case. It has been discussed in more detail in chapter 3 from equation 4.1 to 4.6.

```
[WebMethod] // Method to add square of each character counted in service case and service
request and separates the comma from each value
      public double SqrAddCmaSprStr(string Values){
      double Square = 0;
      string[] ValCh = Values.Split(',');
      try {
      for (int i = 0; i < ValCh.Length; i++) {
      Square = Square + double.Parse(ValCh[i]) * double.Parse(ValCh[i]); }     }
      catch (Exception exptn)    {
      Square = 0;        }
      return Square;     }
```

## 4.6 User Interface(UI)

Purpose of UI is to make interaction of users very easy, simple and efficient in order to complete their goals [36]. Before sending service request users need to register themselves and then they can search the hotel.

To start click on **Enter** button .The main start up screen is shown in figure 24.

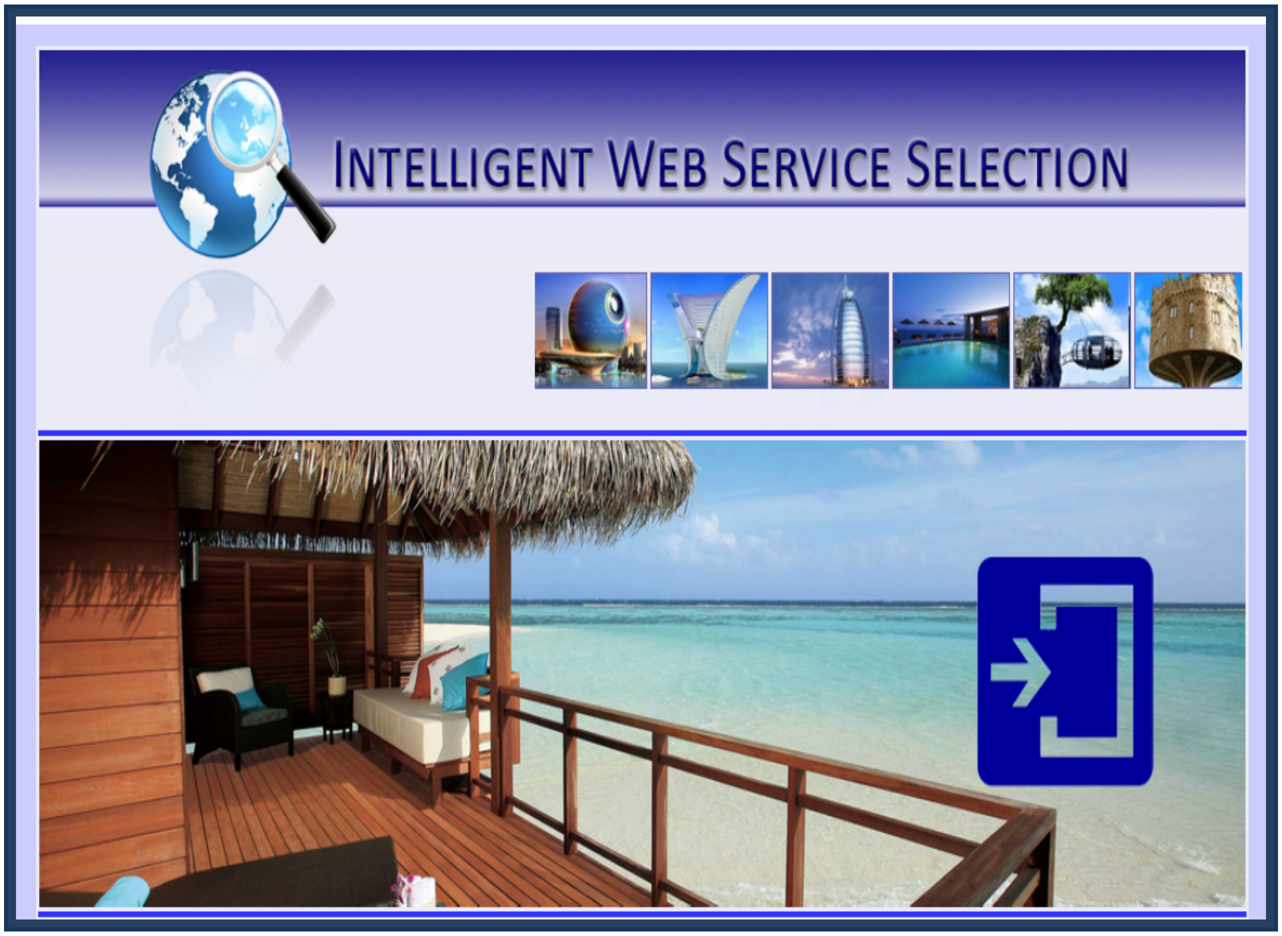
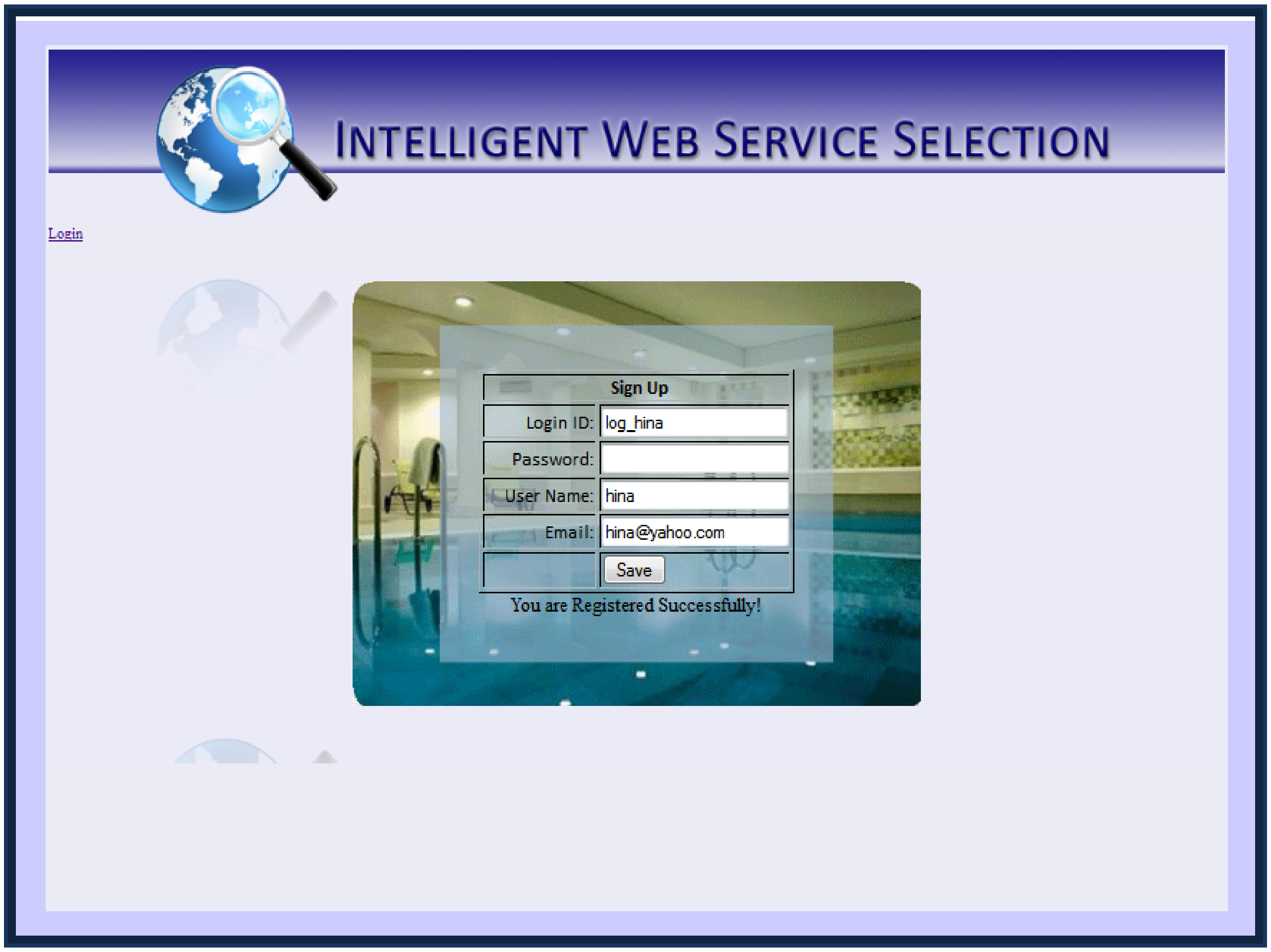


**Figure 24 : Main Start up Screen**

After starting application **Sign Up** page appears as every service consumer needs to register in order to use web service. User enters related information to complete registration and clicks on save button , all data will be saved and user will be registered successfully.



|  |  |
|---|---|
| Sign Up |  |
| Login ID: | log_hina |
| Password: |  |
| User Name: | hina |
| Email: | hina@yahoo.com |
|  | Save |

You are Registered Successfully!

**Figure 25: User Registration Screen**

To login into the system click on **Login** link appearing at the top left corner of sign up page.

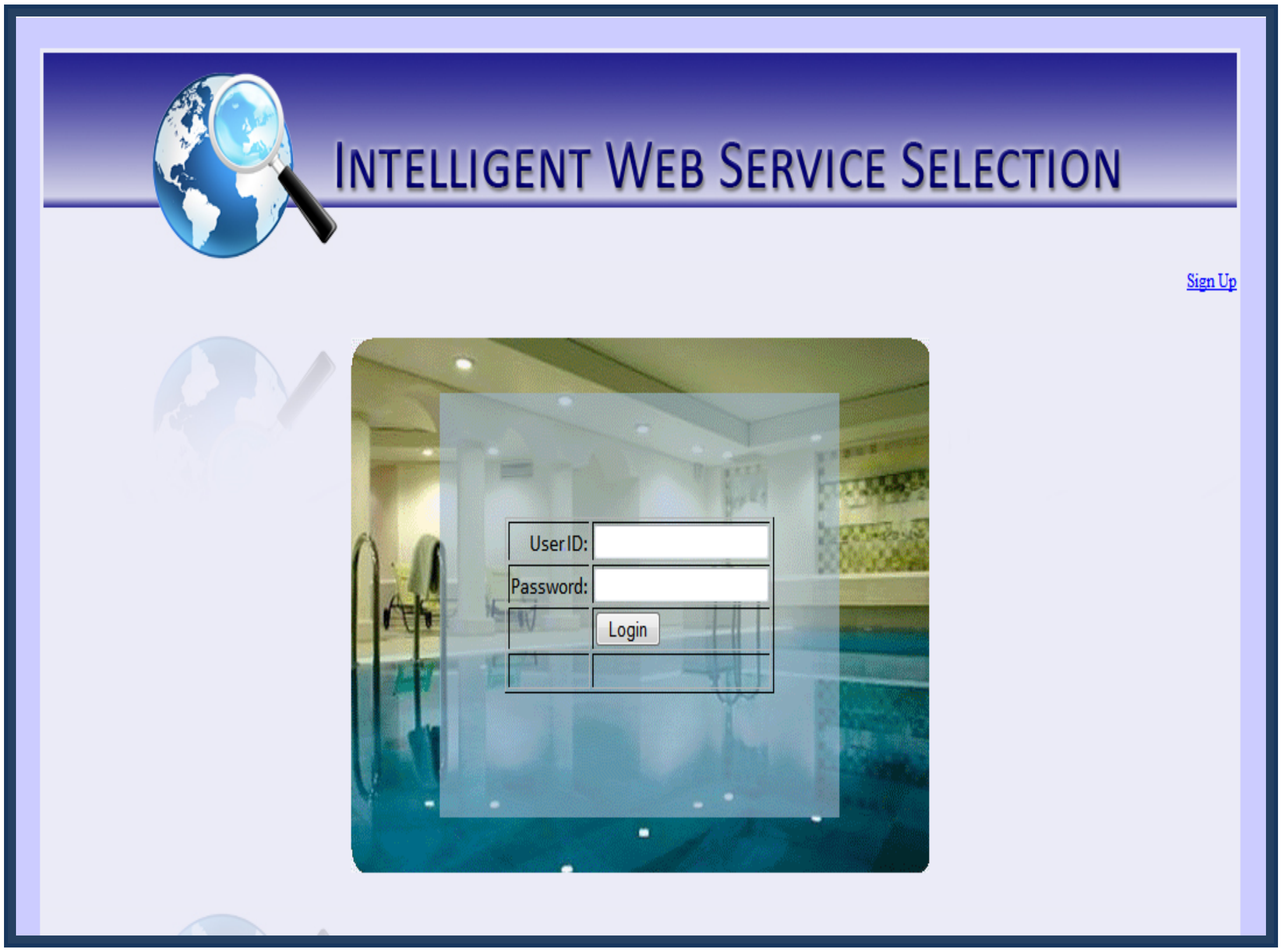


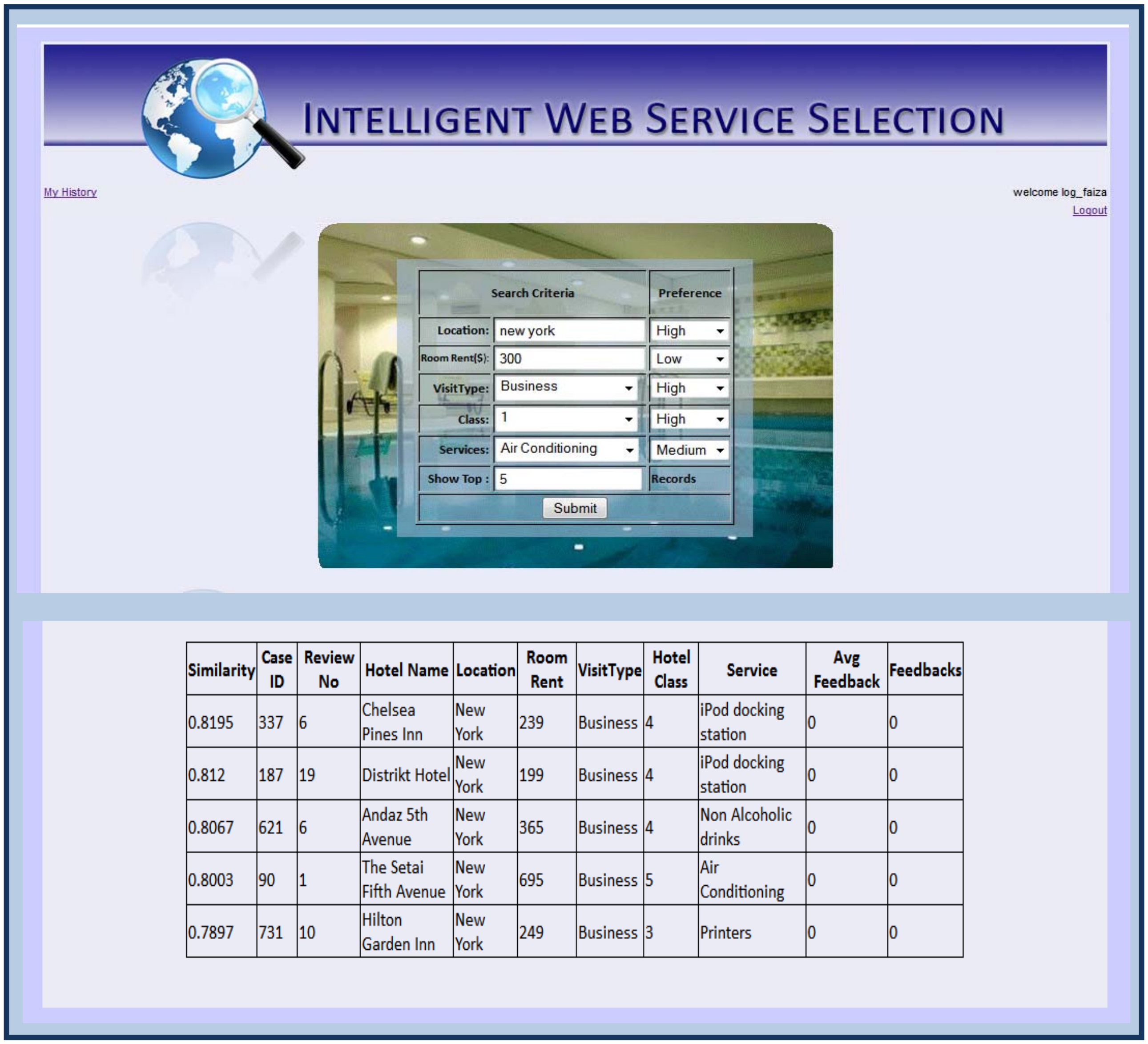
**Figure 26: User Login Screen**

After successfully login user is navigated to **Search Page** as shown in figure 27, where he/she sends the service request by providing five parameters along with preferences and clicks on submit button on the screen and then on the basis of these parameters and their preferences, service cases are returned to user with highest computed similarity at the top.

User's request consists of following parameters:

**Location:**

This field takes non-numeric value. User enters country name where he wants to search hotel.



| Search Criteria | | Preference |
|---|---|---|
| Location: | new york | High |
| Room Rent($): | 300 | Low |
| VisitType: | Business | High |
| Class: | 1 | High |
| Services: | Air Conditioning | Medium |
| Show Top : | 5 | Records |

| Similarity | Case ID | Review No | Hotel Name | Location | Room Rent | VisitType | Hotel Class | Service | Avg Feedback | Feedbacks |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.8195 | 337 | 6 | Chelsea Pines Inn | New York | 239 | Business | 4 | iPod docking station | 0 | 0 |
| 0.812 | 187 | 19 | Distrikt Hotel | New York | 199 | Business | 4 | iPod docking station | 0 | 0 |
| 0.8067 | 621 | 6 | Andaz 5th Avenue | New York | 365 | Business | 4 | Non Alcoholic drinks | 0 | 0 |
| 0.8003 | 90 | 1 | The Setai Fifth Avenue | New York | 695 | Business | 5 | Air Conditioning | 0 | 0 |
| 0.7897 | 731 | 10 | Hilton Garden Inn | New York | 249 | Business | 3 | Printers | 0 | 0 |

**Figure 27: Search Service Screen**

**Room Rent:**

It takes numeric value. User enters the room's rent that he/she can afford.

**Visit Type:**

It takes non-numeric value. There are different types of visit options:

- Solo
- Couple
- Friends
- Business
- Family

**Hotel Class:**

It takes numeric value which specifies the ranking (star) of hotel e.g. 3, 4, 5 and 6.

**Services:**

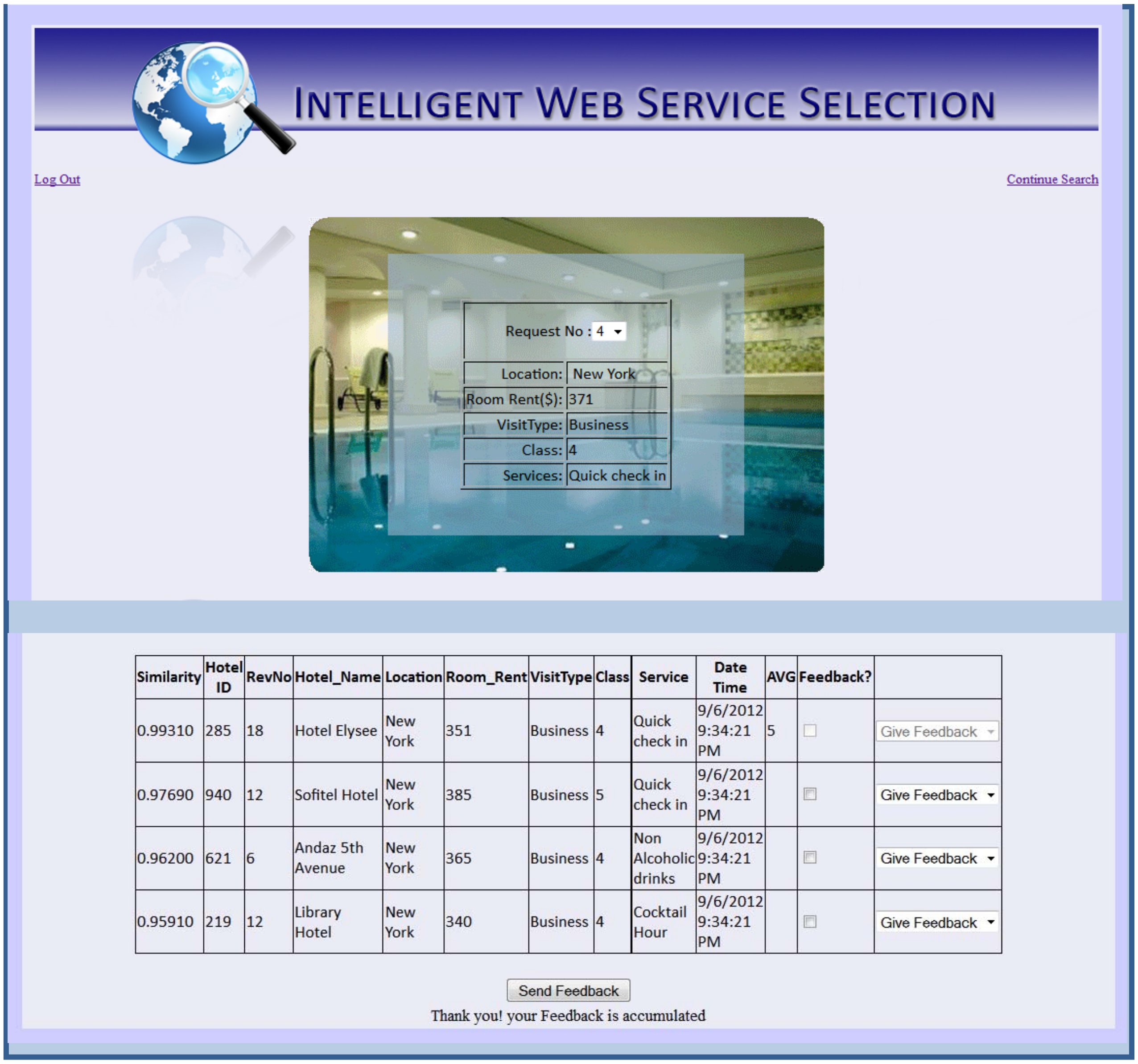
It takes non-numeric value describing the services that user wants to get in that hotel e.g. parking etc.

**Show Top:**

It takes numeric value describing the number of service cases that user wants to be displayed.

Our proposed framework also maintains the history of users' requests so that later on users login to record their experience regarding service in terms of feedback. These feedbacks are provided as Highly Satisfied, Satisfied or Not Satisfied by selecting from drop down whilst against these system stores numeric values and Average feedback is computed by the system. System also saves the date and time when the request was made by particular user.

To record feedback against service request user clicks on **My History** link where he/she can save his/her feedback against service request by selecting desired request number from drop down. System shows their requests and also services that were offered against these requests.



| | |
|---|---|
| Request No : | 4 |
| Location: | New York |
| Room Rent($): | 371 |
| VisitType: | Business |
| Class: | 4 |
| Services: | Quick check in |

| Similarity | Hotel ID | RevNo | Hotel_Name | Location | Room_Rent | VisitType | Class | Service | Date Time | AVG | Feedback? | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.99310 | 285 | 18 | Hotel Elysee | New York | 351 | Business | 4 | Quick check in | 9/6/2012 9:34:21 PM | 5 | ☐ | Give Feedback |
| 0.97690 | 940 | 12 | Sofitel Hotel | New York | 385 | Business | 5 | Quick check in | 9/6/2012 9:34:21 PM | | ☐ | Give Feedback |
| 0.96200 | 621 | 6 | Andaz 5th Avenue | New York | 365 | Business | 4 | Non Alcoholic drinks | 9/6/2012 9:34:21 PM | | ☐ | Give Feedback |
| 0.95910 | 219 | 12 | Library Hotel | New York | 340 | Business | 4 | Cocktail Hour | 9/6/2012 9:34:21 PM | | ☐ | Give Feedback |

Thank you! your Feedback is accumulated

**Figure 28: My History Screen**

Chapter 5

# 5 Results and Discussion

In this chapter we shall discuss the results generated by our proposed framework on testing with test dataset. We have run our system with testing data to achieve the results and then performed analysis of our results. Section 5.1 discusses the data collection. Section 5.2 gives detail of testing dataset. Evaluation measures and parameters are discussed in section 5.3 and 5.4.

## 5.1 Data Collection

We have collected data from TripAdvisor website [29] which is one of the most popular sites providing trip planning services in the form of hotel booking, travel advice, and vacation planning. Data about 25 top hotels in New York was collected including Functional parameters (e.g. prices) and QoE parameters (e.g. hotel services as described by reviewers who had experienced the hotel).

## 5.2 Testing Dataset

We have enriched our database with 924 service cases out of which 100 are used as testing dataset and run our system with this testing dataset. Each case in the test set was in-turn used as a query for the case base and the system returned a ranked list of top N most similar cases.

## 5.3 Evaluation Measures

Our evaluation measure is Precision.

### 5.3.1 Precision

Precision is measured as the fraction of relevant retrieved cases to all the retrieved cases in response to each query in test data. Precision is calculated as below which varies in value from 0 to 1:

$$Precision = \frac{No.of\ relevant\ cases\ retrieved}{No.of\ retrieved\ cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots 5.1$$

Whereas, a threshold value of .89 is used for relevance determination.

## 5.4 Performance Evaluation

The performance of our proposed framework is evaluated by using precision as discussed in section 5.3. To check the performance of our system we have run our test dataset first only with Functional parameters and then with all parameters (Functional and QoE); analyzed the results of both.

### 5.4.1 Calculating Average Precision with Proposed Framework

We have computed Average precision by running 100 queries (i.e. our test dataset) one by one and firstly we provided all the parameters (Functional and QoE) and retrieved top 10 service cases and then continued to increase the set of retrieved service cases to 20, 30, 40,50,60,70 and finally 80; figuring out the number of relevant cases retrieved for these sets and then computed average precision for these sets. Table 3 shows the average precision with our proposed framework.

**Table 3: Average Precision by Proposed Framework**

| N Top Cases | Average Precision by Proposed Framework |
|---|---|
| 10 | .97 |
| 20 | .92 |
| 30 | .88 |
| 40 | .83 |
| 50 | .79 |
| 60 | .74 |
| 70 | .7 |
| 80 | .66 |

We have mapped the above results on graph in figure 29.The horizontal axis shows number of cases retrieved from the case base (N), vertical axis shows average precision value at each test dataset.

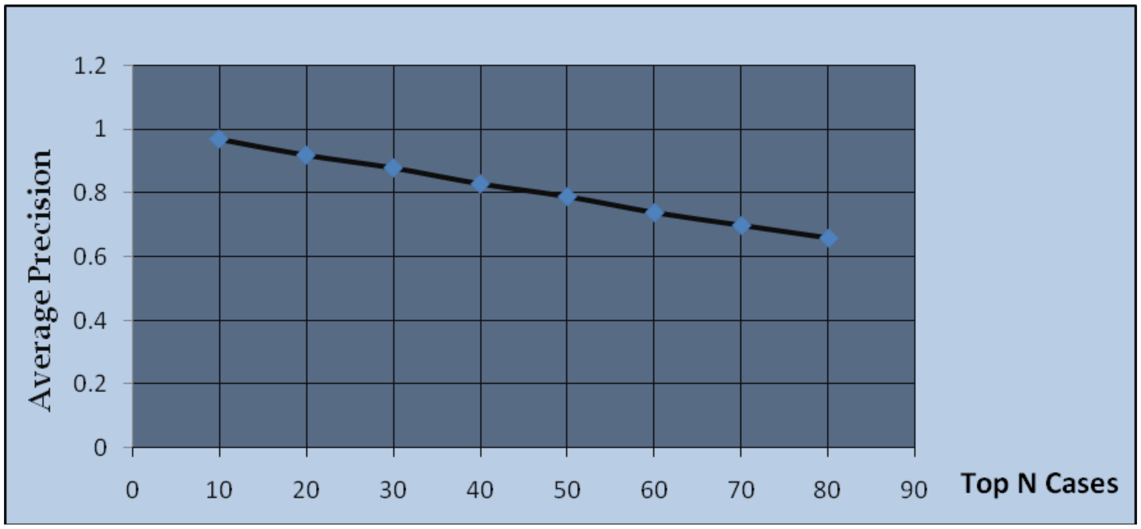


**Figure 29: Average Precision of Proposed Framework**

### 5.4.2 Comparison of Average Precision with Other Technique

Our technique is compared with one other technique "Searching and Selecting Web Services Using Case Based Reasoning" which is proposed by Olivia Graciela Fragoso Diaz [19]. The analysis of the result shows that our proposed technique has improved the precision.

Diaz has considered only the Functional parameters while dealing with user's request for searching the service. To make comparison of our technique with Diaz's technique we have also computed Average precision by running 100 queries (i.e. our test dataset) one by one by providing only Functional parameters and retrieved top 10 service cases and then continued to increase the set of retrieved service cases to 20, 30, 40,50,60,70 and finally 80 as we did in case of providing all parameters (our proposed technique); figuring out the number of relevant cases retrieved for these sets and then computed average precision for these sets.

**Table 4: Comparison of Average Precision of Diaz Framework with Proposed Framework**

| N Top Cases | Average Precision with Functional parameters by Diaz [19] | Average Precision by Proposed Framework |
|---|---|---|
| 10 | .82 | .97 |
| 20 | .6 | .92 |
| 30 | .45 | .88 |
| 40 | .34 | .83 |
| 50 | .28 | .79 |
| 60 | .23 | .74 |
| 70 | .2 | .7 |
| 80 | .17 | .66 |

It is clear from the graph as shown in figure 30 that system performance has improved through our proposed framework by combining the QoE parameters with Functional parameters to search for required services. Table 4 shows the comparison of results.

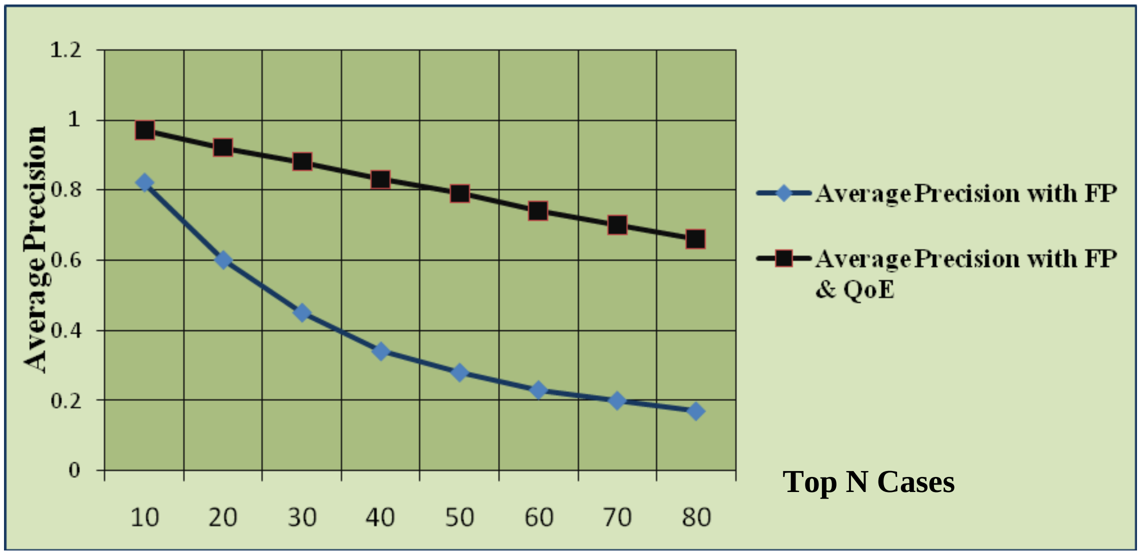


**Figure 30: Precision Comparison with other Technique**

### 5.4.3 Calculating Average Precision with Preferences

We have also performed experimentation by using the preferences with service request in our proposed framework as shown in Table 5. The analysis of the result shows: by incorporating the preferences with Functional and QoE parameters (i.e. our proposed framework) the average precision improves. Table 6 shows the tabular comparison between the average precision with and without preferences by using Functional and QoE parameters.

**Table 5: Average Precision with Preferences in Proposed Framework**

| N Top Cases | Average Precision with Preferences in Proposed Framework |
|---|---|
| 10 | 1.00 |
| 20 | .99 |
| 30 | .97 |
| 40 | .95 |
| 50 | .93 |
| 60 | .91 |
| 70 | .89 |
| 80 | .87 |

**Table 6: Comparison of Average Precision with and without Preferences in Proposed Framework**

| N Top Cases | Average Precision without Preferences in Proposed Framework | Average Precision with Preferences in Proposed Framework |
|---|---|---|
| 10 | .82 | .97 |
| 20 | .6 | .92 |
| 30 | .45 | .88 |
| 40 | .34 | .83 |
| 50 | .28 | .79 |
| 60 | .23 | .74 |
| 70 | .2 | .7 |
| 80 | .17 | .66 |

We have mapped the values of table 6 on graph to show the comparison of results as depicted in figure 34.

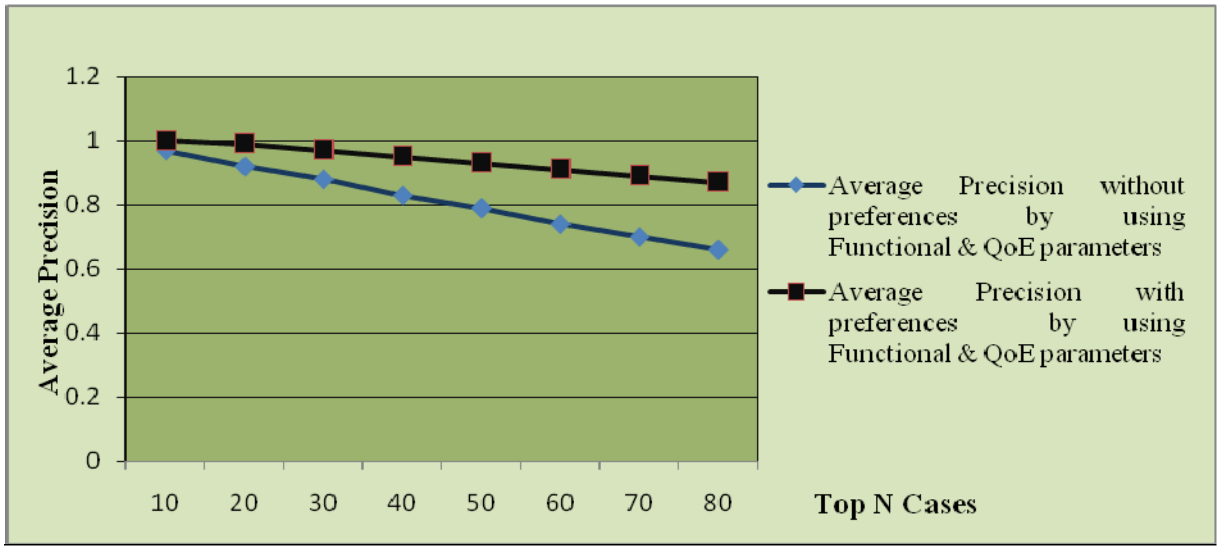


**Figure 31: Precision Comparison with and without Preferences**

### 5.4.4 User Profiles

We have randomly selected few service cases and each case in the test set was in-turn used as a query by considering the preferences provided for each parameter of the service case .The system returned a ranked list of top N most similar cases and average similarity is computed. We have also performed experimentation by using three user profiles:

- Business Profile
- Personal Profile
- Family Profile

We have defined some preferences for these profiles' parameters as shown in Table 7.

**Table 7: Preferences for User Profiles**

| Profiles | Price | Service | Hotel Class |
|---|---|---|---|
| **Business** | Low | High | High |
| **Personal** | High | Low | Low |
| **Family** | High | High | Medium |

#### 5.4.4.1 Calculating Average Similarity for User Profiles:

We have computed average similarity for each query against these profiles as shown in Table 8. Where Q1,Q2,…..,Q6 shows the queries(service cases) that are run for three profiles for top 20 service cases  by considering the preferences as defined in table 5.

**Table 8: Average Similarity**

| Query(Cases) | Business Profile | Family Profile | Personal Profile |
|---|---|---|---|
| Q1 | 0.91 | 0.91 | 0.94 |
| Q2 | 0.95 | 0.93 | 0.90 |
| Q3 | 0.92 | 0.97 | 0.97 |
| Q4 | 0.98 | 0.20 | 0.59 |
| Q5 | 0.86 | 0.92 | 0.96 |
| Q6 | 0.94 | 0.94 | 0.94 |

Figure 32 shows average similarity for family, business and personal profiles.  Vertical axis shows Average similarity and Queries (cases) are shown along Horizontal axis. Analysis of results shows the average similarity of each query for each profile. Suppose for query Q5 the average similarity for business profile is .86, for family profile its .92 and for personal profile its .96 it means same query for different user profiles has different average similarity depending upon the priorities which are specified for different profiles.

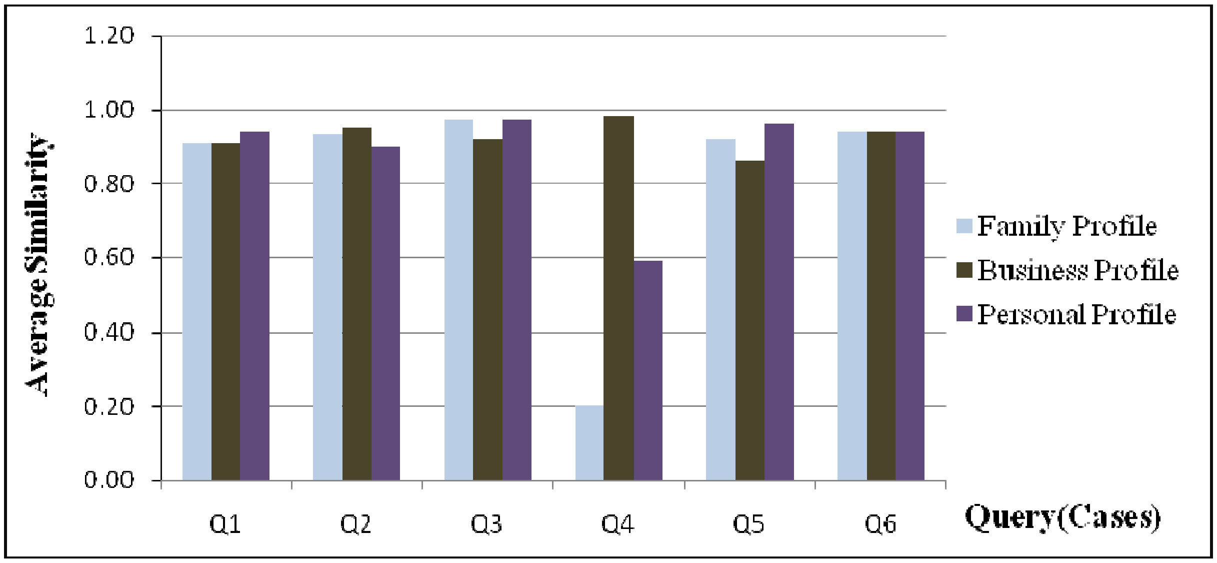


**Figure 32: Average Similarity for Profiles**

Chapter 6

# 6 Conclusions and Future Work

In this chapter section 6.1 discusses overview of research. Objectives achieved through this research are discussed in section 6.2 and 6.3 gives detail about conclusion and future work.

## 6.1 Overview of Research

With increase in number of users towards WS for their information needs, and a proportionate number of available web services, there is a strong demand from consumer community for powerful tools to pick the best service(s) from the available candidates. Web Service Selection Problem (WSSP) leads to the issues involved in selecting the most appropriate service for addressing specific user's needs. The problem is dense not only due to the large number of service parameters, but also due to variations in users' preferences within these parameters. For instance one user may prefer efficiency over price, and another may have the opposite preference. Given that manual solutions to WSSP are not sufficient, models are needed that on the one hand could represent the parameter-to-service mapping decision realistically, and on the other lend themselves to efficient execution.

Therefore, we have developed an OWSS framework by using CBR which provides optimal services to requesters by considering both functional and QoE parameters of service request and utilizing the past experience of service consumers .Whereas, current automated approaches for WS selection do not incorporate QoE for selecting web services during evaluation in response to a user query. Moreover, it provides user control over search in the form of preferences. Our proposed framework is intelligent in terms of problem solving by focusing on experiential knowledge and feedback from service consumers is recorded to provide better services in future.

## 6.2 Objectives Achieved

In this thesis, we have introduced a framework by using CBR and incorporating users' preferences and developed a prototype for IWSSF. An algorithm is proposed which takes both functional and QoE parameters of user's service request and provides optimal services to requesters by considering past experience of service consumers which is provided in terms of feedback and automatically learns from this experience. Better solution based on past experience and user's preferences is provided. Furthermore, more control to users is supplied in terms of preferences for their service request. Finally, the proposed framework archives the experience/knowledge after the problem is effectively solved to assist users to solve similar reoccurring problems in the future.

## 6.3 Conclusion & Future Work

Our mechanism for the selection of web services by using CBR system which is intelligent in terms of problem solving by focusing on experiential knowledge. It saves the existing problems along with their solutions in the form of cases and when new request/problem arrives then it reuses its past experience and retrieves the case with highest similarity by considering user preferences. Moreover, it is also efficient as the manual comparison of user's request with number of available services and selection of desired service out of these is quite time consuming task, but by implementing the proposed framework this task can be performed efficiently and effectively. Currently we are explicitly collecting feedback from users, but further research is recommended to automate this feedback collection mechanism. Furthermore, data mining techniques can be used to enhance service case selection process when we have too much data (service cases) in CBR system.

# 7 Bibliography

[1]. Ray Kurzweil, (1990), The Age of Intelligent Machines, Published by Massachusetts Institute of Technology Press, ISBN 0-262-11121-7 / 0-262-61079-5

[2]. Artificial Intelligence, http://www.encyclopedia.com/topic/artificial_intelligence.aspx,last accessed on 13th June, 2012

[3]. Web Services Architecture Requirements: Technical Report, http://www.w3.org/TR/2002/ WD-wsa-reqs-20020429, last accessed on 2nd June, 2012

[4]. Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua,Philippe Comte, Pel Krogdahl, Dr Min Luo and Tony Newling, (2004),Patterns: Service- Oriented Architecture and Web Services, Published by IBM Redbooks, ISBN 978-0-7384-5317-0

[5]. Mike P.Papazoglou, Service -Oriented Computing: Concepts, Characteristics and Directions,Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE -03),USA, 2003

[6]. Haluk Demirkan, R Kauffman, J Vayghan, Hans-Georg Fill, Dimitris Karagiannis and P Maglio, Service-oriented technology and management: Perspectives on research and practice for the coming, Published in Journal of Electronic Commerce Research and Applications(ECRA-08),vol.7,December 2008

[7]. Hao He, What Is Service-Oriented Architecture, http://www.xml.com/lpt/a/1292, last accessed on 2nd June , 2012.

[8]. D. Richards, M. Sabou, S. van Splunter and F.M.T. Brazier, Artificial Intelligence: a Promised Land for Web Services, Published in Proceedings of 8th Australian and New Zealand Intelligent Information Systems Conference (ANZIIS2-03), pp. 205-210.

[9]. Web services: Benefits, challenges, and a unique, visual development solution: White Paper, April, 2006.

[10].Ernesto Damiani, SOAP message structure, http://ra.crema.unimi.it/turing/materiale/admin/corsi/SOA/materiale/SOA3.pdf, last accessed on 13th June, 2012

[11]. Justin R. Erenkrantz , Web Services: SOAP, UDDI, and Semantic Web, May 2004.

[12]. Verena Henrich, Erhard Henrich, Marie Henrich and Thomas Zastrowd , Service Oriented Architectures :From Desktop Tools to Web Services and Web Applications.

[13]. I Watson, Applying Case-Based Reasoning: Techniques for Enterprise Systems, (1997), Published by Morgan Kaufmann Publishers, ISBN 1558604626

[14]. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.

[15]. SankarK.Pal and Simon C.K.Shiu,Foundations of Soft Case-Based Reasoning,(2004), Published by John Wiley & Sons, Inc, ISBN 0-471-08635-5

[16]. Ramon López de Mántaras and Enric Plaza, Case-Based Reasoning: An Overview

[17]. G. Vadivelou, E. IIavarasan, R. Manoharan and P. Praveen, A QoS Based Web Service Selection Through Delegation, Published in International Journal of Scientific & Engineering Research (IJSER-11), vol. 2, May 2011.

[18]. Mick Kerrigan, ,Web Service Selection Mechanisms in the Web Service Execution Environment (WSMX), Proceedings of ACM symposium on Applied computing (SAC-06), 2006.

[19]. Aad van Moorsel, "Metrics for the Internet Age: Quality of Experience and Quality of Business"Fifth Performability Workshop, September 16, 2001, Erlangen, Germany

[20]. Olivia Graciela Fragoso Diaz, René Santaolaya Salgado, Ismael Solís Moreno, and Guillermo Rodríguez Ortiz, Searching and Selecting Web Services Using Case Based Reasoning ,Proceedings of the 2006 international conference on Computational Science and Its Applications(ICCSA'06), pp. 50-57.

[21]. Jianqiang Hu, Changguo Guo, Huaimin Wang and Peng Zou,Quality Driven Web Service Selection, Proceedings of IEEE International Conference on e-Business Engineering ( ICEBE 2005), pp. 681 - 688.

[22]. Umardand Shripad Manikrao and T. V. Prabhakar ,Dynamic Selection of Web Services with Recommendation System, Proceedings of the International Conference on Next Generation Web Services Practices (NWESP -05), pp. 117

[23]. S. Kalepu and S. Krishnaswamy, Verity: a QoS metric for selecting Web services and providers ,Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops(WISE -03) , pp. 131 – 139.

[24]. Poulia Adamopoulou, Evangelos Sakkopoulos, Athanasios Tsakalidis and Miltiadis Lytras, Web Service Selection based on QoS Knowledge Management,

[25]. E. Michael Maximilien and Munindar P. Singh, Multiagent System for Dynamic Web Services Selection, Proceedings of 1st Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE -05), July 2005

[26]. G. Vadivelou, E. IIavarasan, R. Manoharan and P. Praveen, A QoS Based Web Service Selection through Delegation, Published in International Journal of Scientific and Engineering Research(IJSER-11),vol 7,May 2011.

[27]. L. Taher and H. El Khatib, A framework and QoS matchmaking algorithm for dynamic web services selection, Proceedings of the 2nd International Conference on Innovations in Information Technology (IIT-05).

[28]. Mick Kerrigan, Web Service Selection Mechanisms in the Web Service Execution Environment, Proceedings of ACM Symposium on Applied Computing (SAC-06), pp. 1664-1668

[29]. Abhishek Pandey and S.K.Jena, Dynamic Approach for Web Services Selection, Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS-09), March 2009.

[30]. http://www.tripadvisor.com/, last accessed on December 2011.

[31]. Justin R. Erenkrantz , "Web Services: SOAP, UDDI, and Semantic Web",May 2004

[32]. http://en.wikipedia.org/wiki/Microsoft_Visual_Studio, last accessed on 15th June, 2012

[33]. http://www.deitel.com/articles/csharp_tutorials/20051126/csharpWebServices_Part1.html , last accessed on 15th June, 2012

[34]. http://www.west-wind.com/presentations/dotnetwebservices/DotNetWebServices.asp, last accessed on 15th June, 2012

[35]. http://www.beansoftware.com/ASP.NET-Tutorials/Web-Services.aspx, last accessed on 15th June, 2012

[36]. http://en.wikipedia.org/wiki/User_interface_design , last accessed on 15th June, 2012