Advances In Multidisciplinary & Scientific Research
A Multidisciplinary & Interdisciplinary Journal
Vol. 5 No. 3, September, 2019

Framework for a Qos-History Based Automatic Web Service Selection

Oladosu, J. B.[1] & Adegbite, O.[2]
[1]Department of Computer Science and Engineering,
Ladoke Akintola University of Technology,
Ogbomoso, Oyo State.
[2] Department of Computer Science,
Adeleke University,
Ede, Osun State.
E-mail: oluwaseyi.adegbite@adelekeuniversity.edu.ng
Phone: +2348037918201

ABSTRACT

Web service is getting more popular because of its characteristics like platform-independent, loosely coupled, reusability etc. Due to this, many web services have been developed with the same functionality. The purpose of web service selection is to select the best optimal service for a particular task. Selecting the best web service that can satisfy user's requirement becomes a difficult task because selecting the best service should not be based on functional requirement of the users. This research aims at developing a framework based on Quality of Service (QoS) that will address the aforementioned problems and select the optimal web service for a particular task. This research work builds a framework that takes into consideration the supplied functional and non-functional requirement of the users. The evaluation of QoS parameters from multiple sources such as service provider, history-based record and user rating record.

Keywords: Web service, Quality of Service (Qos), History-based record

Aims Research Journal Reference Format:
Oladosu, J. B. & Adegbite, O. (2019): Framework for a Qos-History Based Automatic Web Service Selection.
Advances in Multidisciplinary Research Journal. Vol. 5. No. 3, Pp 53–66
Article DOI: dx.doi.org/10.22624/AIMS/V5N3P6

1. INTRODUCTION

In recent times, many software development approaches have been proposed by different researchers. Some techniques showed to be essential that it appear in relatively all the approach. One of the most important is modularization and reuse. During analysis phase in software development, complex problem is broken down into sub-problems called modules to simplify the understanding of the problem. Also one of the most recent software development methodology is Service Oriented Architecture. Service Oriented Architecture is becoming a model for software development due to its flexibility, cost effectiveness and adaptability. Service Oriented Architecture is an architectural paradigm and discipline that may be used to build infrastructures enabling those with needs (consumers) and those with capabilities (providers) to interact via services across disparate domains of technology and ownership. It is the most popular approach for building robust network applications.

SOA presents an approach for building distributed applications that are delivered to either other services or end-user application. It provides a flexible connection between applications by presenting every application or resource as a service with standardized interface for communication, enabling them to exchange structured information. SOA requires that services are platform independent and interoperable. Web services are the application of SOAs using the Web (the internet) as the platform for delivery. Web service (WS) is a software component, that provides function services via a standard protocol and can be reused many times in different purposes (Sara, Alsayed & Amany, 2017).

The Web lends itself very well to the service requester/provider model because the Web is fundamentally comprised of groups of services (Web sites / Web applications). Web services can be considered one of the primary current methods to implement SOA. The main advantage of using a Web service is its ability to create standard interfaces, which are externally focused. This means that rather than building (or rebuilding) systems using SOA concepts to enable inter-application or inter-system communication; you can leave existing systems communicate as they already do, and use a Web service as the external interface between the different systems.

This has the effect of allowing systems to communicate in a standard and distributed fashion. Research in web services includes many challenges ranging from service registration to service mining. Service registrar can only support web service discovery based on functional requirement of the user request. The purpose of web service selection is to select the best web service for the current task. Recently selection of web service has not been limited to functional requirement due to the fact that there may be thousands of services offering the same functionality. If service selection is based on only on their functionality, there is high probability that the same service will be selected every time and a poor quality service may be selected. There is need for additional technique in web service selection so as to distinguish the good services from the bad one.

Qualities of service (QoS) of the web service is used to distinguish the functionally similar web services. Service providers specify the values for these properties in service description. These values cannot be trusted. Most service providers specify overrated values which will later affect the performance of the web service. There is need for proper evaluation of the QoS properties. This evaluation helps to rank the services in order to select the best services that will satisfy client functional and non-functional requirements. Some web service selection techniques assume zero weight value for non-functional parameter unspecified by the user when making a request thereby resulting in the selection of poor quality service. Some web selection techniques do not recognize user's (service consumer) preference in selection appropriate service for their request. They base the selection on functional and non-functional properties of the discovered web service alone.

In this research, we propose a model to address the aforementioned problems by selecting the best service for the service consumer considering the user's preference, deriving weight for unspecified QoS parameters and also multiple data source for QoS values.

2. RELATED WORK

Gobinath and Revathi (2013) use particle swarm optimization technique for service selection. They used Service QoS information set which is composed of several QoS ternary constraint relations. Based on this relation they select services and rank service according to weight given by user. Ludwig and Reyhani (2007) apply QoS metrics in Grid computing in order to assure dynamic service selection. They take into consideration execution duration, execution price, reputation, reliability, and availability. The services are ranked according to the QoS criteria and the service is chosen by a match-making or a heuristic algorithm. Yang, Dai, Zhang and Gao, (2005) proposed a QoS driven dynamic selection of composite web services, which takes account of both the QoS properties and interface matching degree. Genetic algorithm optimized neural network algorithm is proposed for the decision to select the best web service.

Sha, Shaozhong, Xin and Mingjing, (2009) proposed a QoS-based web service selection model. This method is based on weightage and normalization of functionally similar services. Though this method gets the client's weightage, there is a large difference between the QoS values of a group of functionally similar services and QoS value of another functionally similar service. During normalization the difference between the normalized QoS values becomes negligible. Thus user's weightage does not influence the QoS values while ranking the web services.

Ilavarasan, Vadivelou, and Prasath, (2008) propose a standardized web service ping operation into all web service. A web service selection architecture, the Delegation web service selector which offers a better solution for implementing web service load balancing and also increase security of web service compared to other approaches.

Liu, Yun-Xiang, Zhang, Tang, and Jing (2007) present an algorithm GODSS (global optimal of dynamic Web services selection) to resolve dynamic Web services selection with QoS global optimal in Web services composition. The essence of the algorithm is that the problem of dynamic.

3. METHODOLOGY

Service selection is a very complex and challenging task, especially if it takes a variety of different non-functional properties into account. The fundamental issues of service selection are (1) specifying requestor's service requirements, (2) evaluation of the service offerings, and (3) aggregating the evaluation results into a comparable unit (Hong and Stephan, 2007). The requestor requirement and service offering falls with the functional requirement while the aggregating the evaluation result falls under the non-functional requirement.

3.1 QoS Criteria For Web Service

Important aspect of web service representation is the advertising of the functional and non-functional ability of the service. Five quality criteria are selected according to their importance to web service and they are discussed below:

1. **Execution time**: measure the amount of time taken by the provider to execute a request. The time duration from service request to response.
2. **Availability**: the probability that a service is accessible. It is the ratio of available time to total time. Total time is composed of available time and down time. 1-down time/total time.
3. **Price**: the amount of money the service consumer as to pay for executing the request of the service provider.
4. **Reliability**: is the ratio of successful requests to all the request. Success means that a consumer get the correct information they requested for.
5. **Reputation**: it's a measure of the web service trustworthiness. This parameter depends on the end user's experiences of the web service.

3.2 The Framework

Figure 3.1 shows the new system processes of the framework for a QoS-History based automatic web service selection. Web service quality model based on Quality of Service parameters that are applicable to all web service were adopted. The final objectives is to obtain a cost function that will be used to rank the services that fulfill the functional requirement and non-functional requirement (QoS) of the service requestor and select the web service with the lowest cost function as the best web service among the ranked web services.

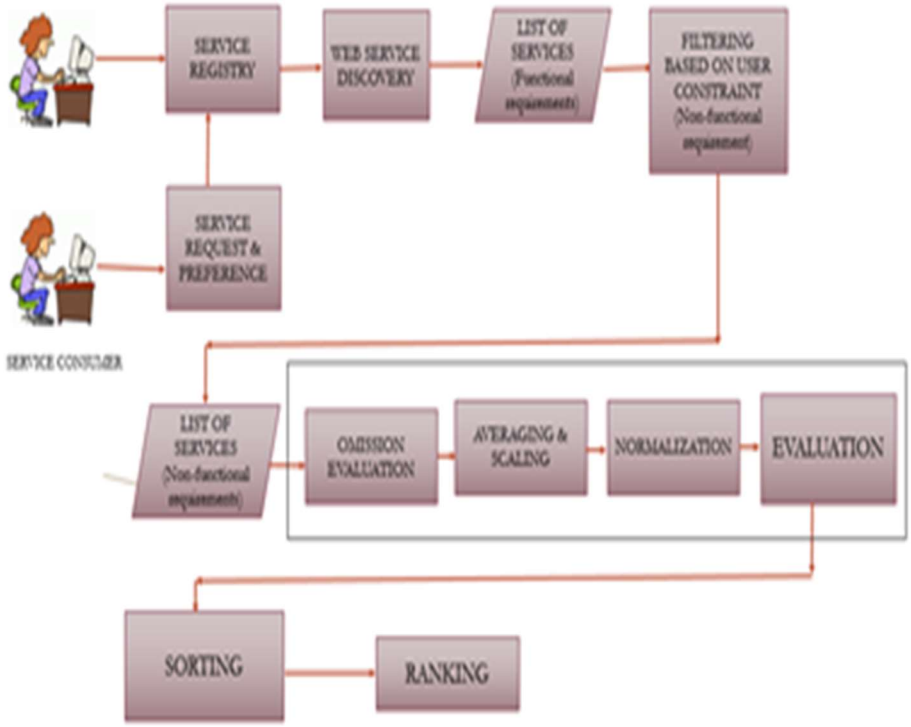

Figure 3.1: Steps in Selection the Best Web Services.

In Fig. 3.1, represent all the steps in selecting the best web service among the candidate services that can satisfy the user's functional and non-functional requirement. Service provider publish their web service to the service registry. They also provide additional information which include functionalities, WSDL, QoS etc. The service consumer request for service from the service registry with its non-functional preferences and constraints.

3.2.1 Service Discovery

Web services provide functionalities by their input and output functions. The input function is carried out on input parameters. These parameters are needed for the functionality of the method within the system. In order to find the best web service, it is necessary to find the services in the service registry that can satisfy the functional need of the consumer. Service discovery is important in web service selection. Service discovery is the procedure of finding appropriate web service that can satisfy user's functional needs. This is done by searching the registry for the web services. A large list is compiled from the service registry. The list of candidate web services that satisfy service consumer request is gathered during service discovery.

$S_{(f)} = \{s1,s2,s3,s4,\ldots,sn\}$ ……(1)

$S_{(f)}$ represent the generated list of candidate web services that satisfy functional requirement of the service consumer.

Figure 3.2: Web Service Discovery

3.2.2 Omission Evaluation

Some service consumer may be indifferent with some QoS parameters, so thereby leaving them unfilled when requesting for a web service that will satisfy their functional and non-functional needs. There is need to set a default value for all the QoS parameters should incase the service request left the values unfilled. This will prevent the application from assuming a value 0 for uninitialized QoS parameter.

3.2.3 Filtering

After service discovery, the first thing to do is to check if the discovered web services satisfies the user's QoS preferences and their corresponding constraints. Constraints are defined on the parameters. For example, the consumer may request for a service with execution time less than 100millisecond and reputation to be 10. The list of the web services is filtered to delete those services that cannot fulfill the non-functional preferences and constraints of the service consumer.

$S_{(f,k)}$={s1, s2,s3,..,sn} ……(2)

K represent constraint, f represent functional requirement, $S_{(f,k)}$ represent the list of the generated candidate web service that satisfy the functional and non- functional constraint.

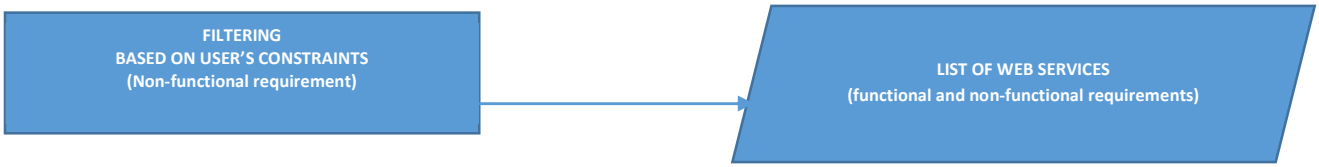

Figure 3.3: Web Service Filtering Based On Functional and Non-Functional Requirements

3.2.4 Averaging And Scaling

Average and scaling is the procedure to evaluate the relative and average value of each QoS parameter in the service community. This will help in evaluating for the weight of each web service. In order to compute the weight of each candidate web service in the generated list, the average and relative values of QoS parameters needs to be computed. The relative value for each QoS parameter will be computed in relation to the average value. It is computed by dividing by dividing the actual QoS value of the web service with the corresponding average value. Average value of each QoS parameter in the service community is the sum of all the parameters divide by the number of candidate web services.

$I_{av} = (\sum_1^n I)/n$ (3)
$I_{rel} = I/I_{av}$ (4)
$I_{aval} = (I_{aval\_reg}+I_{aval\_log})/2$ (5)
$I_r = (I_{r\_reg}+I_{r\_log})/2$ (6)
$I_e = (I_{e\_reg}+I_{e\_log})/2$ (7)
$I_p = I_{p\_reg}$ (8)
$I_{rep} = (I_{rep\_reg}+I_{rep\_rating})/2$ (9)

Where I_{av} is the average value of each QoS parameter, n is the total number of qualified candidate web services in the service community, I_{rel} is the relative value of each QoS parameter, I_{aval}, I_r, I_e, I_p and I_{rep} represent QoS parameters availability, reliability, execution time, price and reputation respectively and I_{aval_reg}, I_{r_reg}, I_{e_reg}, I_p and I_{rep_reg} represent each QoS parameter value provided by the service provider respectively and I_{aval_log}, I_{r_log}, I_{e_log} and I_{p_reg} represent QoS values from the execution log database and I_{rep_rating} is the QoS value from the user feedback database.

3.2.5 Normalization

We need to normalize different non-functional properties. Some of the parameters are of minimum value i.e. the higher the value, the lower the quality. Example include execution time. While the other some others are of maximum value i.e. the higher the value, the higher the quality. Examples include reliability, reputation, and availability price.

$N_i = W_i * N_{rel}$ ---------- negative
$N_j = W_j / P_{rel}$ ----------positive

Where N_{rel} and P_{rel} represent the relative value for the positive and negative parameters, W_i and W_j represent the user preferences.

3.2.6 Evaluating Cost Function

To find the best web service that satisfy both functional and non-functional requirement of the consumer, we need to calculate the cost function of each of the web service. The cost function is the weighted function that represents the total costs of invoking a web service. The value is calculated from the QoS parameters and the service consumer's preferences. Combining all the above equations together, the cost function of a web service is defined as

$F_{(s)} = w_1/(I_p/I_{av}) + w_2/(I_r/I_{av}) + w_3/(I_{rep}/I_{av}) + w_4/(I_{aval}/I_{av}) + w_5*(I_e/I_{av})$
Where $F_{(s)}$ is the cost function, w_1, w_2, w_3, w_4 and w_5 corresponds to user's preference for each parameter.

3.2.7 Sorting & Ranking

For any given task, the system will choose the web service that satisfies all the user constraint and have minimal cost function. After calculating for the cost function of each candidate web services, the web services are sorted in ascending order, the least cost function as the first in the list. The web service with the least cost function will be selected as the best web service. The smaller the cost function the better the service.

3.3 Architecture

The architecture has to:

i. Collect user's preference and constraints
ii. Set a default value for un-specified non-functional parameter
iii. Normalize the QoS parameters
iv. Measure the QoS after the execution of the web service and store it to the execution log database
v. Calculate the cost function of the web service

The architecture is composed of:

1. Service provider: they publish web services to the registry
2. Service consumer: they request for a particular service
3. Execution log database: it stores QoS parameters for past web service execution
4. User feedback database: it stores user rating of the executed web services

4. IMPLEMENTATION

Implementing an automatic web selection framework that can select the best web service among the candidate web services that can satisfy service consumer functional and non-functional needs depends on four main components: Service registry, Service Consumer/Client, Execution log database, User's feedback database. The components are saddled with the following responsibilities: register web service, store their information, measure the QoS parameters, calculate the cost function, sort the web services, rank the web services, choose the best web service base on the lowest cost function.

4.1 Service Registry

Each web service is registered in the database of the service registry. The service registry collects information relating to each of the web service. The information include: service id, execution time, availability, price, reliability, reputation, service name, service address, and service method as shown in Table 4.1. In the database, the information are collected at the point of registering the web service.

4.2 Service Consumer/Client

The client makes a request for a web service in the service registry. The request of the client is the functional needs of the service consumer. It also specify the non-functional preference and constraints on the parameters. After the best web service must have performed the required task, the service consumer is required to give a feedback. Service consumer can rank the web service according its judgment. The reputation value is stored in the user feedback database.

4.3 Execution Log Database

The database keeps record of the past history of each web service. Once a web service respond to a request from the service consumer the execution information are updated in the database.
The following are the QoS parameters stored in this database: execution time, availability, as shown in Table 4.2.

4.4User's Feedback Database

The database stores the ranking of the executed web service as experienced by the service consumer. This values are updated after every web service response to a request.

4.5 DISCUSSION

Series of experiments were conducted to investigate the effectiveness of the new framework. Selecting the best optimal web service among candidate web service for any given functional and non-functional needs. Initially web service selection was based on the advertised information in the service registry. With the increase in service request the history-based record increased and the selection is now according to multiple data source. All experiment were conducted Dell inspiron intel® Pentium ® with a 1.90GHz CPU and 4.00GB. The algorithm was implemented using PHP, phpmyadmin for the database and wamp as the server. QoS parameters were generated randomly except for execution time. The generated QoS parameters are stored in the database. Word encryption was used in evaluating the performance of the framework. 7 encryption web services were used in the experiment. Below are the tables from the service registry, execution log and user feedback database.

Table 4.1: QoS data from the service registry

Id	time	availability	price	reliability	reputation	name	address	method
1	238.09	89	70	73	6	encrypt_crypt	encrypt_crypt?wsdl	crypts
2	226	85	90	73	9	encrypt_bcrypt	encrypt_bcrypt?wsdl	bcrypt
3	229.23	89	30	73	7	encrypt_md5	encrypt_md5?wsdl	md5p
4	253.42	98	75	67	2	encrypt_des	encrypt_des?wsdl	des
5	249.27	87	29	73	8	encrypt_hash	encrypt_hash?wsdl	hashs
6	253.04	80	50	67	8	encrypt_hashsalt	encrypt_hashsalt?wsdl	hashsalt
7	372.34	98	45	67	5	encrypt_passhash	encrypt_passhash?wsdl	passhash

Table 4.1 shows the details of the database of service registry. The data are the information supplied by the service provider during service registration.

Table 4.2: QoS data from the execution log database

id	Time	availability	reliability
1	268	100	100
2	298	99	94
3	226.78	99	87
4	272	97	84
5	223.87	65	73
6	235.35	100	100
7	376.79	52	50

The QoS values in the execution log database as shown in Table 4.2 are the values from the execution log after executing a particular request from the service consumer.

Table 4.3: QoS data from the rating database

id	Rating
1	1
2	8
3	9
4	6
5	6
6	1
7	9

4.6 Service Selection Changes Depending On The Non-Functional Constraints

Three requester (X,Y,Z) were considered with the same functional need but different non-functional preferences and constraints. The functional need for all the requesters is encryption of a word while the non-functional requirements are execution time, availability, price, reliability and reputation. The values of the non-functional requirement of each requester varies.

Scenario A

The REQUESTER X request to encrypt a particular word with the following non-functional constraints:

i. Execution time – 254
ii. Availability – 80
iii. Price – 28
iv. Reliability – 67
v. Reputation – 2

It can be deduced from the non-functional values supplied by the client that the execution time is the client preferences. The cost function calculation take into consideration the execution time than other QoS parameters. The service with ID 1 has the lowest cost function, thus it is selected as the best web service to fulfill the service consumer request. The ranked services are shown in the table below:

Table 4.4: cost function of the candidate web services for scenario A

ID	NAME	COST FUNCTION
1	Encrypt_crypt	422
6	Encrypt_hashsalt	427
2	Encrypt_bcrypt	428
3	Encrypt_md5	428
4	Encrypt_des	436
5	Encrypt_hash	464

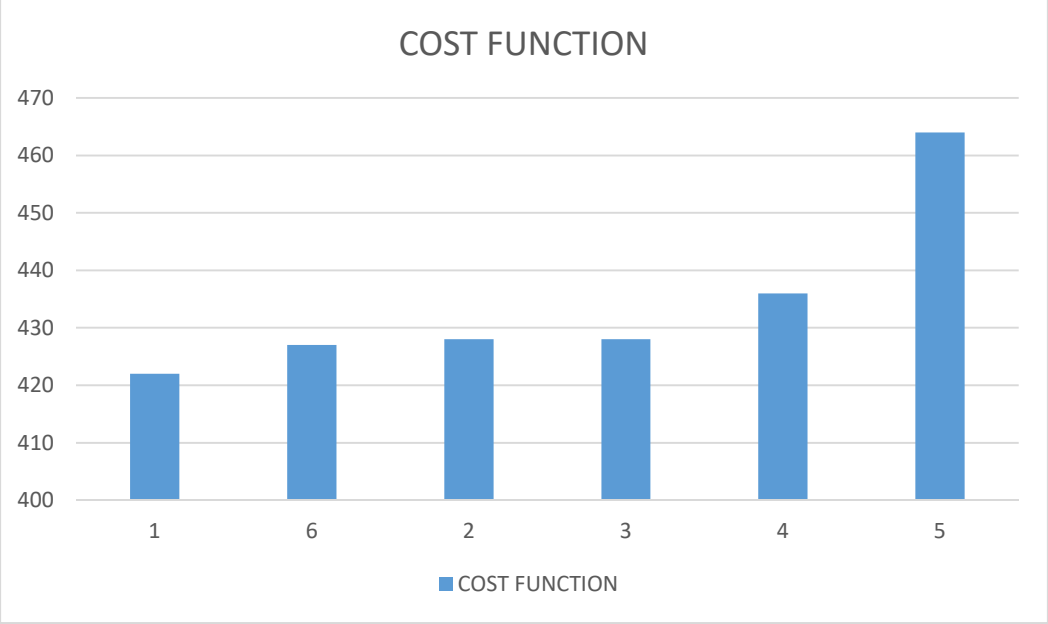

Figure 4.1: chart showing the cost function of Scenario A.

4.5.2 User Qos Parameters Omission On Service Selection

Two requester (A,B) were considered with the same functional requirement but with omission in non-functional preferences and constraints. The requester left some QoS values unfilled while making the request for the best web service for encryption.

Scenario I:

REQUESTER A with the following non-functional constraints:

i. Execution time – 250
ii. Availability – 0
iii. Price – 28
iv. Reliability – 60
v. Reputation – 0

It can be infer from the QoS values enter by the service consumer that reputation and availability of web services is not the user preference. Consideration will be given more to other QoS parameters.

Table 4:6 cost function of the candidate web services for scenario I.

ID	NAME	COST FUNCTION
2	encrypt_bcrypt	317
1	encrypt_crypt	335
3	encrypt_md5	354
5	encrypt_hash	377

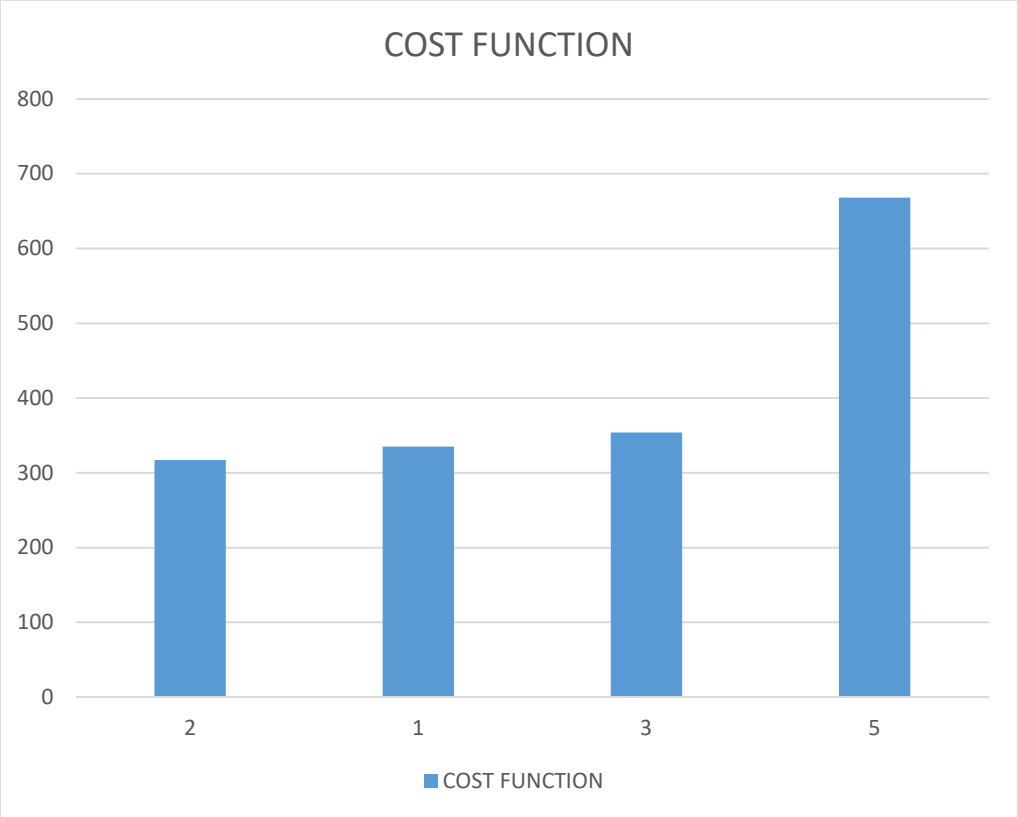

Figure 4.3: chart showing the cost function of Scenario I.

SCENARIO II:
REQUESTER A with the following non-functional constraints:

i. Execution time – 250
ii. Availability – 89
iii. Price – 0
iv. Reliability – 60
v. Reputation – 0

The service requester preference are execution time, availability and reliability, it is indifferent when it comes to price and reputation.

Table 4:7 cost function of the candidate web services for scenario II.

ID	NAME	COST FUNCTION
3	encrypt_md5	394
1	encrypt_crypt	404

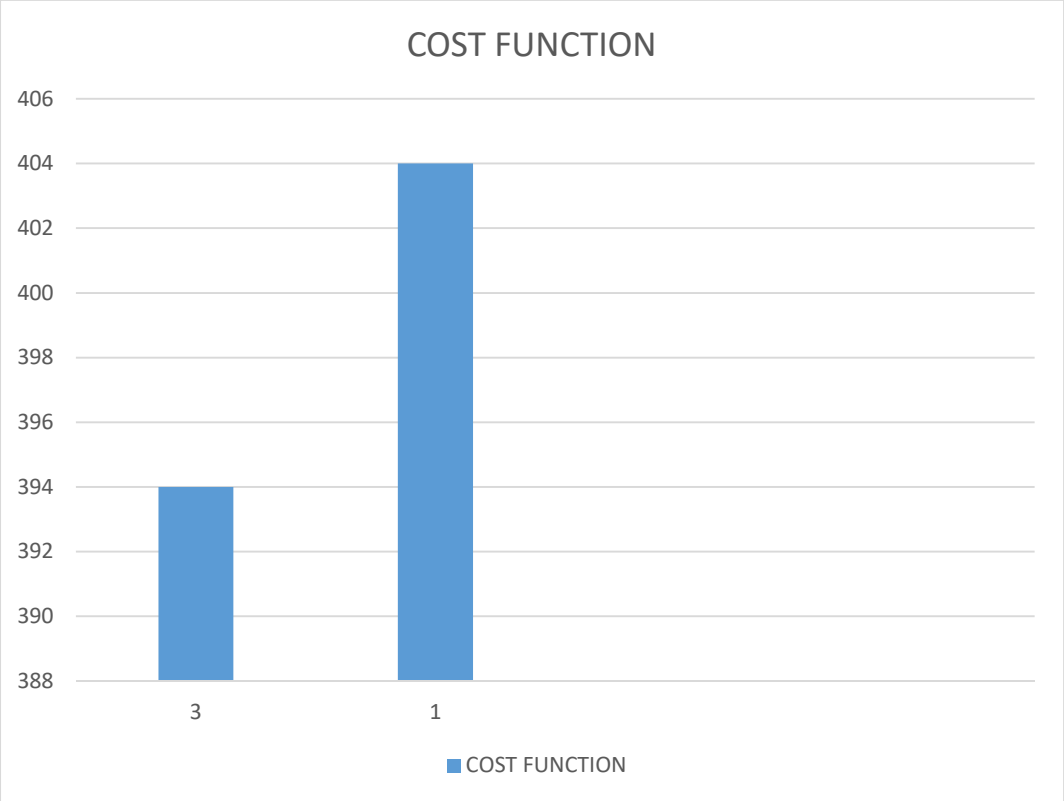

Figure 4.4: chart showing the cost function of Scenario II.

4.5.3 Performance Evaluation With Qos Based Selection Algorithm (QBSA)

The output of the new framework was compared with the output of QoS based selection algorithm (QBSA) proposed by Abdulmotaleb and Deyang (2009) which also used cost function in selecting the best web service for a service consumer.

The following QoS were used in evaluating the performance and effectiveness of the new framework:

i. Execution time – 451
ii. Availability – 80
iii. Price -29
iv. Reliability – 67
v. Reputation – 5

Service with ID 2 was returned from both approaches with cost function of 559 and 607 respectively. This shows that service with ID 2 is the most effective as this aggress with the output in QBSA. The case shows that the new framework is better than QBSA.

Table 4.8: Ranking And Cost Function Of The Candidate Web Services Of The New Framework

ID	NAME	COST FUNCTION
2	Encrypt_bcrypt	559
1	Encrypt_crypt	583
3	Encrypt_md5	596
6	Encrypt_hashsalt	631
5	Encrypt_hash	633
7	Encrypt_passhash	826

Table 4.9: Ranking And Cost Function Of The Candidate Web Services Of QBSA

ID	NAME	COST FUNCTION
2	Encrypt_bcrypt	607
5	Encrypt_hash	624
3	Encrypt_md5	634
6	Encrypt_hashsalt	640
1	Encrypt_crypt	705
7	Encrypt_passhash	861

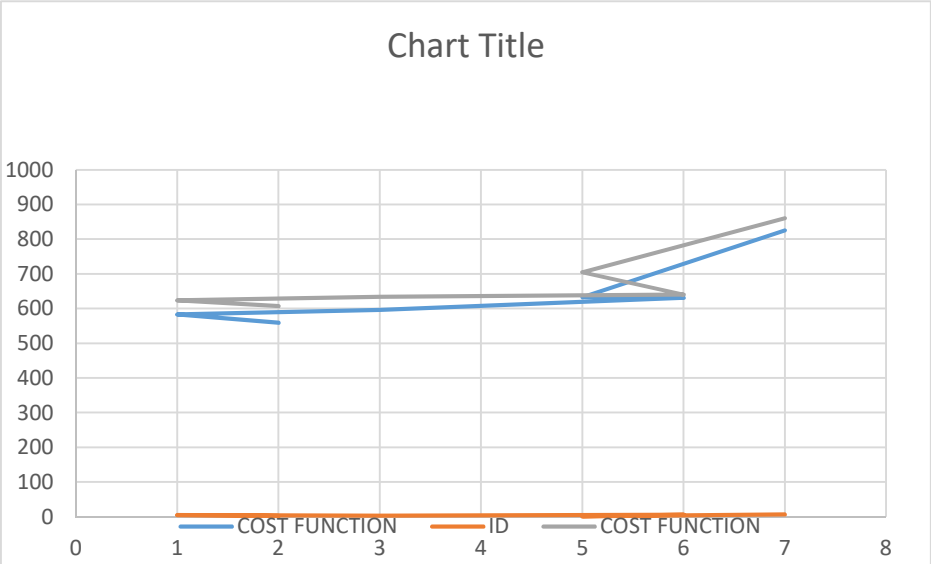

Figure 4.5: Chart Showing the Comparison Between The New Framework and QBSA

5. CONCLUSION

This project covers the aspect of web service selection based on service consumer functional and non-functional requirement and also taking into consideration the QoS constraints. The study shows that the new framework is effective for the selection of the best web service among candidate services in the service community. The implementation was done on local machine, future work should implement the framework on published online web service. Other parameters also affect the selection of web services like security, penalty, and accuracy. Further research can focus on these areas.

REFERENCES

1. Gobinath, J., and Revathi, D. (2013). “Performance View of Knowledge Based Quality of Web Service”. International Journal of Advanced Research in Computer Science and Software Engineering, 3(4), 446.
2. Ilavarasan, E., Vadivelou, G., and Prasath, S., (2008). “Dynamic Selection of Web Services”. Retrieved from https://pdfs.semanticscholar.org/9da1/8a5492860fc36658688888b4cda10fbec81e.pdf
3. Liu, D., and Ralph, D. (2009). “The reverse c10k problem for server-side mashups”. In Service-Oriented Computing – ICSOC 2008 Workshops, pages 166–177.
4. Liu, Y., Ngu, A., and L. Zeng (2004). “QoS Computation and Policing in Dynamic Web Service Selection”. In proceedings of the 13th International Conference on World Wide Web (WWW’04), New York, NY, USA, pp. 66–73.
5. Ludwig, S., and Reyhani, S. (2007). “Selection Algorithm for Grid Services based on a Quality of Service Metric”. 21st International Symposium on High Performance Computing Systems and Applications (HPCS’07) p. 7.
6. Sara, S., Alsayed, A., and Amany, S.,(2017). “Context-Based Web Service Discovery Framework with QoS Considerations”. 11th International Conference on Research Challenges in Information Science (RCIS)
7. Sha L., G. Shaozhong, C. Xin, and L. Mingjing (2009). “A QoS based web service selection model”. Proceedings of the International Forum on Information Technology and Applications (IFITA '09), pp. 353–356.
8. Yang, Y., Dai, B., Zhang, Y., and Gao, Y. (2005). “Dynamic selection of composite web services based on a genetic algorithm optimized new structured neural network, Proc”.IEEE Conference on Cyberworlds.
9. Peter, R (2015). “Introduction to Soap, WSDL and UDDI, the Combo For Big Web Services”. Retrieved from http://indigoo.com