Reduction of Uncertainty in Optimization of Web Services using Dynamic Proxy Method

Aishwarya . D
M.Tech
Department of CSE
Pondicherry Engineering College

Thirumagal.E
M.Tech
Department of CSE
Pondicherry Engineering College

Balachandar.K
M.Tech
Department of CSE
Pondicherry Engineering College

R. Rajkumar
M.Tech
Department of CSE
Pondicherry Engineering College

ABSTRACT
Service orientation has become the most significant technology for businesses to reach their customers in an efficient manner. For end users sake B2B era of web is getting changed to B2C era of web by introducing possible support methodologies to the customers. One of the main factors that have to be taken into consideration in this B2C era of web services is quality of services (QoS). Different optimization methodologies were introduced in order to improve the performance of the services during its delivery to the clients.(Application level). But we argue that – every optimization methodology has some amount of uncertainty in measuring the QoS of the services and in specific no defined methodology for QoS management is provided at the end user access level. Hence we use a dynamic proxy which directly detects the QoS of the services during run time and select the best among them for the end users. This dynamic proxy helps in selecting the best service according the user interests thereby increases the accessibility, utility and robustness of the web service domain.

Keywords
Web Services, QoS, Functional Factors, Non – Functional Factors

1. INTRODUCTION
Optimization of service systems has become an important issue and it needs more attention. Optimization of services can be done at two levels. The primary level includes QOS collection, matching of QOS, monitoring and binding of services. The secondary level includes optimization of services at end user level (i.e.) application level QOS management. It is essential that the services should be deployed in a better way at the client side after the SLA agreement of the user. Till now, no good amount of concentration is given to a quality attributes of the services and to the improving methodologies of the same. The proposed system introduces dynamic proxy method to access web service which replaces the static stub method of accessing web service in the existing system. By using dynamic proxy method it becomes easier to the end user where the proxies are generated at run time and don't have to regenerate static stubs whenever new methods are added to the web service. QOS parameters such as execution time, availability, reliability, execution cost and other such parameters can be considered to optimize a service. System optimization reduces the execution time and improves the efficiency. There are several challenges in optimizing a service based on the client's request. First, a given enterprise may have several number of similar services to mash up by searching services and finding its compatibility and finally to provide an optimized service to satisfy the need of client. This work deals with providing optimized service to the client. The optimization occurs at two levels, the primary level includes QOS collection, monitoring and binding/rebinding, transaction support, QOS negotiation and matching. Whereas, the secondary level optimization is the one we concentrate here which includes application level QOS management. This optimization refers to the QOS management of the service at the end user level where the user can access the service using dynamic proxy method. The Dynamic proxies reduce the classes that have to be packed with the application thus improving the accessibility of the user. The work provides various numbers of services where the user can access a service by agreeing the conditions. The user can also register a new service to the existing pool of services. When the user agrees the required service to access, the service gets deployed at the client side automatically where the user can modify the user interface or any other modification with the coding of the service. The Application level management improves optimization of the services, reduces the difficulty in accessing services, runtime proxies are generated which increases the efficiency, dynamic proxies reduces the classes that have to be packed with the application and there is no need of regeneration of static stubs on adding new methods the existing service logics.

2. RELATED WORKS
Web services are gradually becoming a dominant form of granular components used to build applications in today's business IT systems. In a highly dynamic system environment like this, services that can produce similar or complementing results need to be quickly identified in a number of scenarios [2]. If any involved Web services become unavailable for some reason, the whole business process appears unavailable to the clients or partners. For the composition of Web services non-functional characteristics are commonly considered criteria for finding and selecting available services. With aggregated QoS it can be verified whether a set of services satisfies the QoS requirements for the whole composition or not. The aggregation performed builds upon abstract composition patterns, which model basic structural elements

of a composition like parallel paths, a sequence, or a looped execution [5]. The broker supports and integrates the concept of fine-grained access control and scheduling into a single component. Such an integration at the resource scheduling layer avoids situations in which a user finds out that he/she is not authorized to execute his/her job on the resource allocated by the scheduler[7]. There is huge number of web services in the web service group. The web services in the particular web service group will provide the same functionality but with various QoS parameters. This paper proposes the optimization model which is QoS oriented for selecting the web service from the web service group using three steps namely Defining the satisfaction degree of performance, functionality, cost and trust of the web service, Formal description of optimization model, The optimization is done using two searching methods such as optimized clustering using decision tree method and QoS predictions [8]. In existing system the application level QOS management and the mechanisms for selecting a good service system is provided. A general approach to optimizing service systems must include at least robustness, system orientation, and being dynamic and transparent. Robustness The ability to cater to the varying rigidities on Web service QoS in distinct application domains and of various users using a robust solution in a heuristic manner; System orientation The ability to formulate the overall system utility of service systems in the system perspective of a particular system end user and suggest its maximization to that end user; and Being dynamic and transparent The ability to dynamically achieve higher perceived system utility of service systems (again heuristically) by adjusting QoS levels of different components of the system in a transparent way to the system end user. A contribution to the issue of optimizing service systems has been made by presenting a general solution that accommodates the above three key elements. In particular, the work include 1) supporting robust statements such as "If the quality of an application is better than the probability of the expectation which is considered as fulfilled," 2) making it system-utility-enabled to take care of the distinct characteristics of measuring usefulness by the end users, and 3) encompassing dynamic and transparent negotiation to achieve the maximal usefulness of the system for its system end users [1]. An extra layer is introduced in the web service QOS stack to improve the application level QOS management as in Fig 3.1. Robust optimization has been applied to deliver promising results in immunizing uncertainty in optimization against infeasibility while preserving the tractability of the model.

3. DYNAMIC PROXY FRAMEWORK

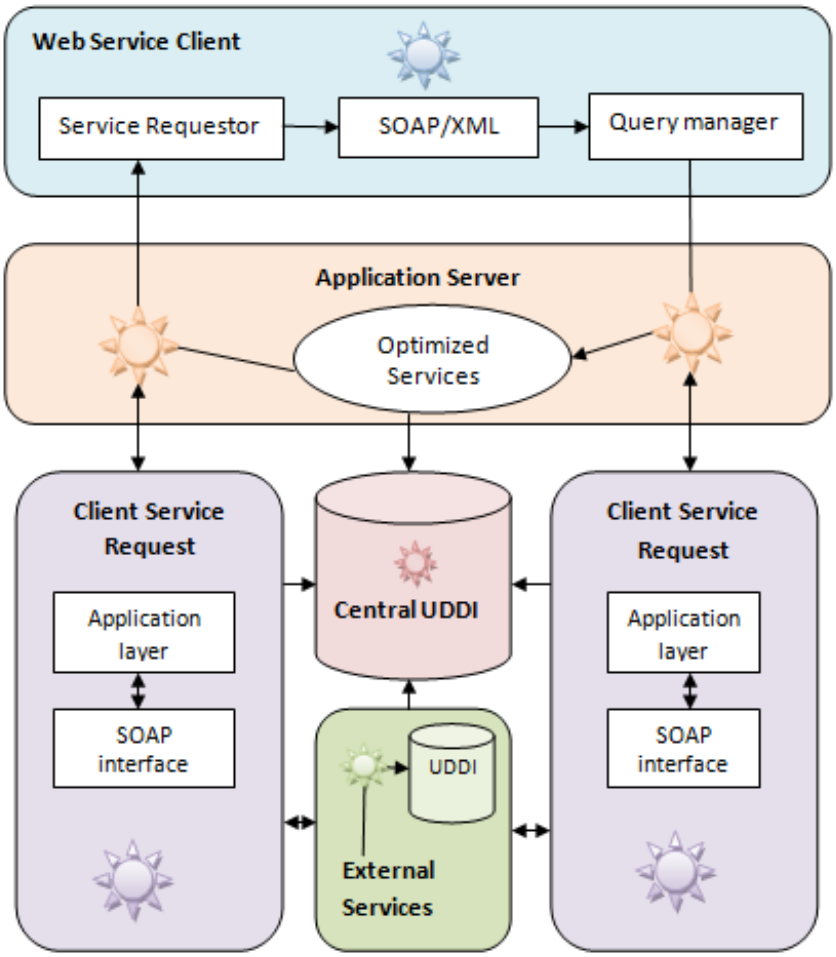

Fig 1: System Architecture for QoS Optimization

The framework proposed in this paper is the effective one for providing the better service access at the end user side. The main purpose is to provide the optimized service at the application level (i.e.) how efficiently and effectively the service available can be accessed at end user application level over the network. In this section, we put forward an implementation framework for collecting and analyzing the service based on the QoS of that service, and then describes our experiment. The experiment describe the process of collecting service based on information and selecting services under the agreement signed by the client to the service provider. Web service client locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its web services. The client also coded in java that uses the Web Service defined in first part. We have to define a dynamic invocation interface client. The Server Container includes a set of Java classes, libraries, and other files that are distributed along with Java server programs. This provides system services that enable a Java server program to execute. Here the server used is the Sun Application Server, which acts as the container. The experiment implements the components of web service in the J2EE platform and uses the Microsoft Access database to store records. A hundred services providing 'By a Container' called the server (Sun Application Server) and function will be deployed in the client accessible platform. Before making e available to the user it is essential to deploy the service at the server side. This page shows how the service has been deployed in the server side. After completing series of steps for deployment, the services will be available in the server. Similar to the deployment at the server side, it is essential to deploy the service at the client side too. After the deployment of the service in the client side, the service consumer can access and use it. If the deployment is successful then the "Operation Completed Successfully" message will be displayed. All the services which are active will be displayed with its details for the easy use of service consumer. The service consumer can get the service by negotiation. The strategic negotiation is done by comparing the values passed to the each attribute. The home page consists of the various functions like service registration, service level agreement, service registry, QoS interface and contact details. The available services are also listed in the home page which are already created and stored in the service registry. The services are listed for the easy use of the consumer. If any of the links is clicked then the user is directed to the respective web service description page. The service registering page consists of all the services that have been already registered in the registry, also provide facilities to register the new service. In order to register the new service it is essential to enter the service name, its response time, execution time, service cost, uptime and its respective WSDL address. The entered details are stored in database and displayed as available services. If the service has been registered successfully then the "Registered Successfully" message will be displayed. After the successful registration of the service, the service will be updated in the database with all the details entered. The database consists of service name, cost, execution time, response time, WSDL and uptime of the service. Before making the service available to the user it is essential to deploy the service at the server side. This page shows how the service has been deployed in the server side. After completing series of steps for deployment, the service will be available in the server. Similar to the deployment at the server side, it is essential to deploy the service at the client side too. After the deployment of the service in the client side, the service consumer can access and use it. If the deployment is successful then the "Operation Completed Successfully" message will be displayed. All the services which are active will be displayed with its details for the easy use of service consumer. The service consumer can get the service by negotiation. The strategic negotiation is done by comparing the values passed to the each attribute. If the client selects any of the service and agrees with all the terms and conditions of service provider, then client can sign the SLA by clicking the agree button available at the bottom of the page. After the agreement of the service required, the service gets registered in the client machine, packed with its classes by displaying a message to the consumer. The dynamic proxy method is used to access the web service where the classes can be easily packed and regenerated whenever changes made at the server. Once after the service has been successfully registered and deployed in the client machine, the service consumer can easily access the service. The service will be available at the client side with all its required classes. The details entered by the consumer while accessing the service is updated in the respective service database

4. EXPERIMENTAL ANALYSIS

The measure of Accessibility, Utility, Robustness and Efficiency of the system is tested using the implemented environment and the results are shown in the following graphs and table.

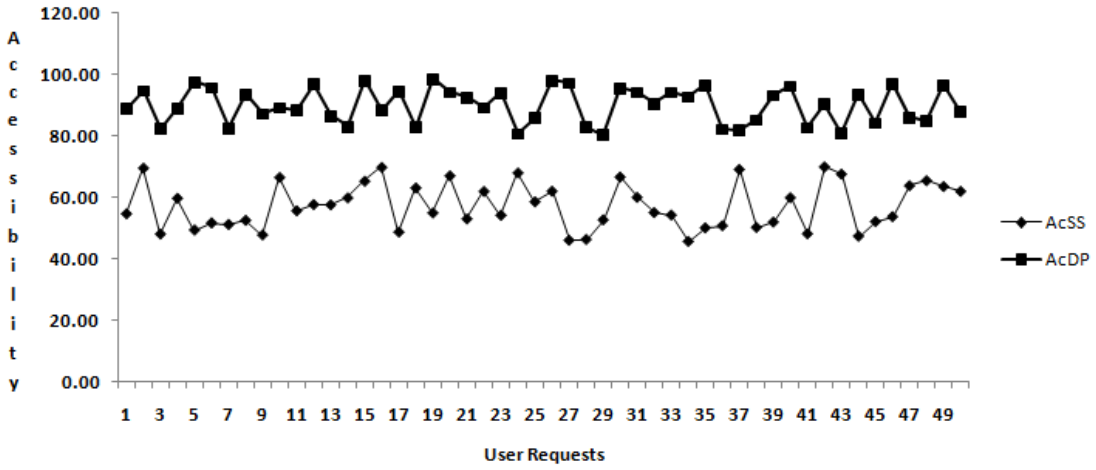

Fig 2: Accessibility using Static stub and Dynamic Proxy

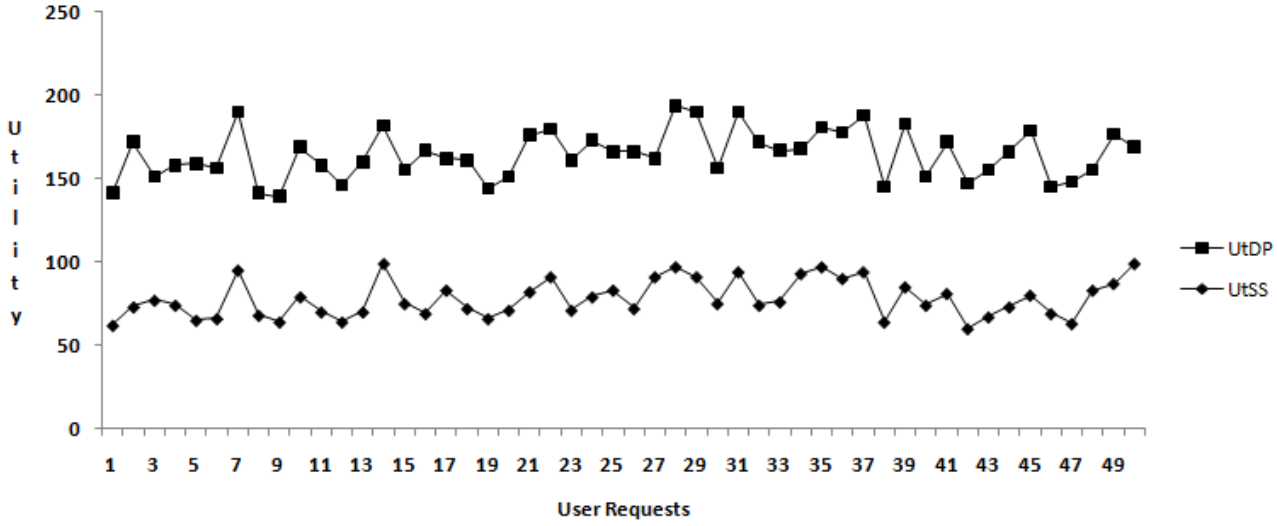

Fig 3: Utility using Static stub and Dynamic Proxy

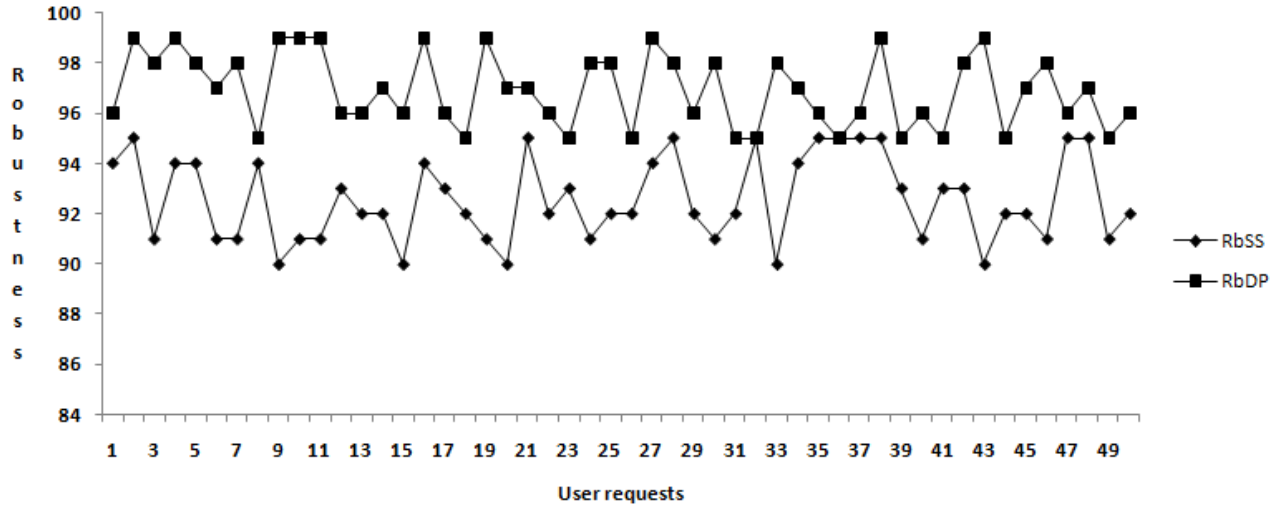

Fig 4: Robustness using Static stub and Dynamic Proxy

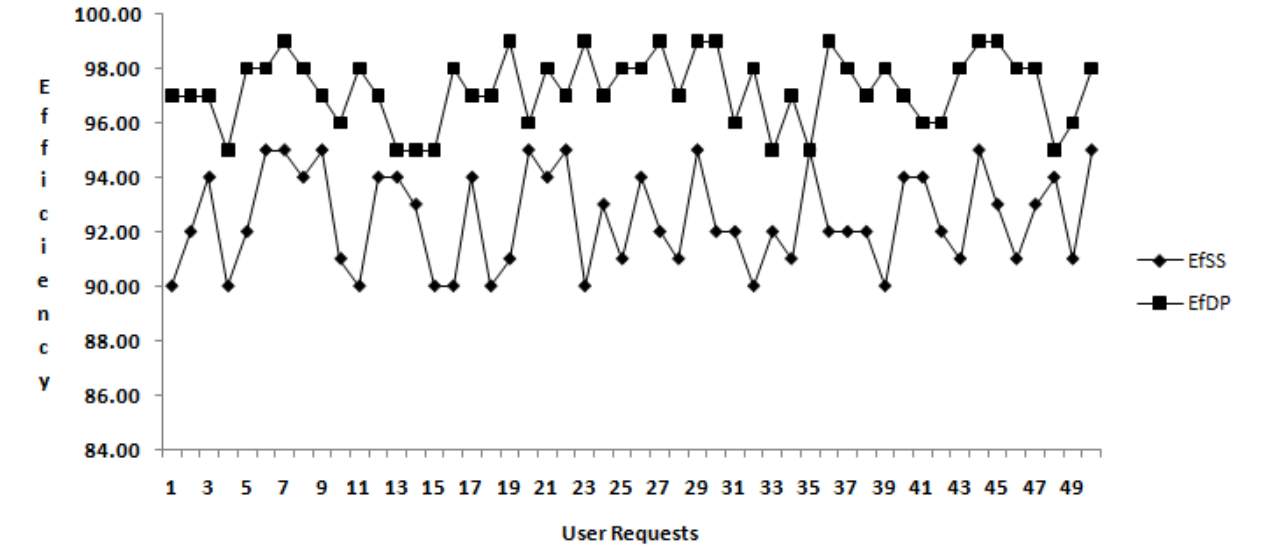

Fig 5: Efficiency using Static stub and Dynamic Proxy

ReqID	AcSS	AcDP	UtSS	UtDP	RbSS	RbDP	EfSS	EfDP
1	56.17	90.06	77.00	84.00	91.00	95.00	91.00	97.00
2	55.32	90.66	83.00	99.00	91.00	98.00	95.00	96.00
3	45.29	84.45	97.00	99.00	92.00	99.00	93.00	98.00
4	56.34	90.17	63.00	99.00	94.00	97.00	95.00	99.00
5	61.04	82.59	98.00	72.00	94.00	95.00	94.00	99.00
6	69.12	89.81	77.00	73.00	93.00	96.00	91.00	97.00
7	64.11	84.63	92.00	87.00	92.00	96.00	91.00	99.00
8	47.68	90.16	97.00	74.00	94.00	96.00	95.00	99.00
9	54.44	88.75	83.00	81.00	90.00	98.00	95.00	96.00
10	50.47	80.27	76.00	86.00	94.00	97.00	95.00	95.00
11	52.16	88.48	63.00	90.00	95.00	98.00	95.00	96.00
12	46.86	84.83	81.00	98.00	91.00	98.00	95.00	97.00
13	55.52	94.55	75.00	98.00	92.00	97.00	93.00	96.00
14	62.82	84.34	73.00	73.00	94.00	95.00	95.00	98.00
15	66.53	84.49	88.00	83.00	95.00	96.00	95.00	96.00
16	56.56	91.03	65.00	97.00	94.00	98.00	93.00	98.00
17	50.32	82.35	96.00	85.00	94.00	96.00	93.00	96.00
18	53.58	96.54	77.00	99.00	90.00	97.00	94.00	98.00
19	68.60	95.97	67.00	72.00	92.00	98.00	90.00	95.00
20	59.20	93.77	65.00	78.00	95.00	97.00	93.00	97.00
21	57.24	95.14	77.00	78.00	91.00	96.00	92.00	99.00
22	69.70	86.67	84.00	92.00	92.00	97.00	94.00	99.00
23	54.35	94.29	87.00	88.00	93.00	96.00	92.00	95.00
24	57.89	92.01	67.00	78.00	90.00	96.00	93.00	99.00
25	47.95	97.60	72.00	88.00	90.00	98.00	90.00	98.00
26	57.03	93.96	94.00	82.00	92.00	96.00	91.00	95.00
27	45.65	93.37	78.00	73.00	91.00	97.00	91.00	97.00
28	48.96	89.23	90.00	83.00	90.00	96.00	93.00	96.00
29	47.20	98.12	94.00	97.00	91.00	99.00	93.00	98.00
30	49.84	88.58	89.00	83.00	92.00	96.00	93.00	98.00
31	52.91	95.90	74.00	95.00	92.00	99.00	91.00	96.00
32	65.50	89.45	66.00	70.00	92.00	99.00	94.00	95.00
33	50.15	93.68	66.00	83.00	94.00	97.00	95.00	98.00
34	66.83	86.59	85.00	96.00	93.00	95.00	94.00	99.00
35	63.68	85.33	96.00	82.00	93.00	98.00	92.00	98.00
36	56.25	84.00	70.00	84.00	95.00	97.00	94.00	96.00
37	57.30	90.64	70.00	91.00	92.00	98.00	95.00	95.00
38	59.79	96.42	71.00	91.00	94.00	95.00	94.00	97.00
39	53.00	98.50	99.00	96.00	94.00	97.00	93.00	97.00
40	47.30	93.79	74.00	90.00	91.00	97.00	92.00	95.00
41	52.60	90.29	75.00	81.00	90.00	99.00	91.00	97.00
42	61.50	98.11	84.00	76.00	94.00	96.00	94.00	97.00
43	53.63	88.93	80.00	76.00	91.00	99.00	95.00	95.00
44	54.03	81.04	67.00	95.00	95.00	98.00	95.00	97.00
45	47.42	80.46	70.00	81.00	93.00	95.00	92.00	97.00
46	65.01	95.64	89.00	80.00	91.00	97.00	95.00	96.00
47	55.09	98.57	87.00	83.00	90.00	95.00	93.00	95.00
48	64.94	87.48	95.00	79.00	92.00	98.00	91.00	98.00
49	53.06	90.56	77.00	97.00	94.00	96.00	94.00	97.00
50	69.77	93.55	89.00	88.00	93.00	95.00	95.00	95.00

5. CONCLUSION

Web service optimization with application level QOS management has reduced greatly user's uncertainty in accessing a service. The existing systems mainly concentrate on the primary level QOS management which includes reliability, accessibility and response time. The application level QOS management at end user level is also an important area to focus and thereby to improve quality of a service being provided. Our proposed system provides the user to register a service available and to access a service required by agreeing the conditions provided. The accessing of a web service is made easier in our system with the help of browser based client which uses dynamic proxy method. By using dynamic proxy method to access service it becomes easier to pack the classes required by the user. Web services are the next step in the web's evolution, since they promise the infrastructure and tools for automation of business-to-business relationships over the Internet. The integration of web services into the J2EE 1.4 platform simplifies the task of building and consuming web services, by freeing the Java technology developer from the low-level details of XML and web services standards. Thus proposed system is enhanced with methods to access a service in a better way reducing the uncertainties.

6. REFERENCES

[1] Qianhui Liang, Member, IEEE,Xindong Wu, Senior Member, IEEE,and Hoong Chuin Lau"Optimizing Service Systems Based on Application-Level QoS "IEEE Transactions On Services Computing, Vol. 2, No. 2, April-June 2009.

[2] Q. Liang, H. Lam, L. Narupiyakul, an P.C.K.Hung," A Rule-Based Approach for Availability of Web Service," Proc. IEEE Int'l Conf. Web Services (ICWS '08), pp. 153-160, 2008.

[3] D.A. Menasce, "Composing Web Services A QoS View," IEEE Internet Computing, vol. 8, no.6, pp. 88-90, Nov./Dec. 2004.

[4] Q. Liang, H.C. Lau, and X. Wu, "Robust Application Level QoS Management in Service Oriented Systems," Proc. IEEE Int'l Conf.e-Business Eng., pp. 239-246, 2008.

[5] M.C. Jaeger, G. Rojec-Goldmann, and G. Muhl, "QoS Aggregation in Web Service Compositions," Proc. IEEE Int'l Conf. e-Technology, e-Commerce, and e-Service (EEE), pp.181-185, 2005.

[6] L.J. Zhang, "EIC Editorial Introduction to the Knowledge Areas of Services Computing," IEEE Trans. Services Computing, vol. 1,no. 2, pp. 62-74, Apr.-June 2008.

[7] P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Bertino,"Efficient Integration of Fine-Grained Access Control in Large-Scale Grid Services," Proc. IEEE Int'l Conf. Services Computing (SCC'05), pp. 77-84. 2005.

[8] S. Cheng, C.K. Chang, and L.J. Zhang, "Modeling and Analysis of Performance Oriented and Revenue Based Admission Control Framework for Service Providers," Proc. IEEE Services Computing Workshops (SCW '07), pp. 9-16, 2007.