An Efficient WS-QoS Broker Based Architecture for Web Services Selection

T.Rajendran
AP cum Research Scholar
Department of CSE
SNS College of Technology

Dr.P.Balasubramanie
Professor
Department of CSE
Kongu Engineering College

Resmi Cherian
Final Year ME (CSE)
Department of CSE
SNS College of Technology

ABSTRACT
Web Service selection is a key component in service-oriented computing. Service-oriented Architectures follow the find-bind-execute paradigm in which service providers register their services in public or private registries, which clients use to locate web services. The QoS based web service selection mechanisms plays an essential role in service-oriented architectures, because most of the applications want to use services that accurately meet their requirements. Currently, the UDDI catalogue supports only primitive matching mechanisms and provides no control over the quality of registered services. We propose a QoS broker based architecture for dynamic web service selection which facilitates the clients to specify the non-functional requirements like QoS along with functional requirements. The paper presents an efficient mechanism for finding the most suitable web service according to the consumer's requirements.

Categories and Subject Descriptors

Primary Classification:
H. Information Systems, H.3 INFORMATION STORAGE AND RETRIEVAL, H.3.5 On-line Information Services, Subjects: web-based services.

Additional Classification:
C. Computer Systems Organization, C.2 COMPUTER-COMMUNICATION NETWORKS, C.2.4 Distributed Systems, Subjects: distributed applications

General Terms
Algorithms, Design, Verification

Keywords
Web Service Selection, Quality of Services (QoS), WS-QoS Broker, UDDI, WSDL, SOAP, tModel

1. INTRODUCTION
A service-oriented architecture is essentially a collection of services that communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Hence some means of connecting services to each other is needed. A key driver for SOA implementations is the hope to save development time and costs through a higher degree of reuse of components in the form of readily implemented services [3],[4]. To achieve this aim it is necessary, among other things, to make Web services discoverable. SOA services have self-describing interfaces in platform-independent XML documents. Web Services Description Language (WSDL) is the standard used to describe the services. SOA services communicate with messages formally defined via XML Schema. Communication among consumers and providers or services typically happens in heterogeneous environments, with little or no knowledge about the provider. SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Discovery, and Integration (UDDI) is the standard used for service registry. Web services are self-described software entities which can be advertised, located, and used across the Internet using a set of standards such as SOAP, WSDL, and UDDI. Web services encapsulate application functionality and information resources, and make them available through programmatic interfaces, as opposed to the interfaces typically provided by traditional Web applications which are intended for manual interactions. However, discovering web services using keyword-based search techniques offered by the existing UDDI registry does not yield results that are tailored to client's needs. Several web services may share similar functionalities, but possess different non-functional properties. When discovering web services, it is essential to take into consideration, the functional and non-functional properties in order to render an effective and reliable service selection process.

Nowadays, both Web Service providers and clients are concerned with the QoS guaranteed by web services. From the client point of view, web service based QoS discovery is a multi-criteria decision mechanism that requires knowledge about the service and its QoS description. However, most of clients are not experienced enough to acquire the best selection of web service based on its described QoS characteristics. They simply trust the QoS information published by the provider; however most of web services providers do not guarantee and assure the level of QoS offered by their web services. Based on the above we propose a Web Services discovery architecture that contains an extended UDDI to accommodate the QoS information, and WS-QoS Broker to facilitate the Web Service discovery.

Measuring the degrees to which the web services can deliver the functionality through a combination of QoS parameters becomes significant, particularly in distinguishing services competing in the same domain. The QoS parameters can be used to characterize the web services' overall behavior. Service providers QoS claims may not be trustworthy. Hence some method is needed to

automate the process of measuring QoS for registered web services. Current UDDI registries don't have built-in-capabilities to validate or monitor published web services. They include only metadata about businesses and their related web services. If the UDDI registries let service providers publish their QoS claims, they could publish false or inaccurate information or the published information could be passive or outdated. Hence the clients should be able to obtain web service information based on QoS metrics from a trusted service broker. This would yield more relevant results.

QoS delivered to a client may be affected by many factors, including the performance of the web service itself, the hosting platform and the underlying network. A set of verification procedures is essential for providers to remain competitive and for clients to make the right selection and trust the published QoS metrics. For the success of any QoS based web services architecture, it should support a set of features: 1) QoS Verification and Certification to guide web service selection 2) QoS aware web services publishing and discovery. In this paper, we propose a broker based architecture for web service selection and QoS management. The role of the WS-QoS broker is to support QoS provisioning and assurance in delivering web services. It implements the concept of QoS verification and certification.

The remainder of the paper is organized as follows. Section 2 outlines the related research conducted in the area of web services discovery, QoS and reputation. In Section 3, we describe the architecture of our proposed WS-QoS broker. Section 4 concludes the paper and presents possible future research in this direction.

2. Related Work

Web service technology uses an interface description to expose its functionality and makes it publicly available for use by other programs. Standard web services protocols such as WSDL and UDDI are designed mainly for their functional features. Such protocols do not provide QoS support and verification. Several web services may have similar functionalities but with different QoS property values. When discovering web services, it is necessary to consider both functional and non-functional properties. But the UDDI registry does not include QoS information. To solve this problem, some work has been implemented for enhancing UBR's inquiry operations by embedding QoS information within the message. An example is the UDDIe [1], which provides an API that can associate QoS information through a set of user defined properties. The search queries are executed based on these properties.

Blum [2] proposed to extend the use of Technical models (tModels), within the UDDI to represent different categories of information such as version and QoS information. Ran [5] proposed an extended service discovery model containing the traditional components: service provider, service consumer and UDDI registry, along with a new component called a Certifier. Certifier verifies the QoS of a web service before its registration. However, it lacks support for the dynamism of web services. Majithia et al [6] proposed a framework for reputation-based semantic service discovery. Ratings of services in different contexts are collected from service consumers by a reputation management system. Rajendran and Balasubramanie [7] proposed a framework for agent-based web services discovery with QoS to select the suitable web service that satisfies the client's preferences and QoS constraints. It contains an extended UDDI to accommodate the QoS information. IBM proposes Web Service Level Agreements (WSLA), which is an XML specification of SLAs for web services focusing on QoS Constraints [8]. Many of these approaches do not provide guarantees as to the accuracy of the QoS values over time or having up-to-date QoS information.

UDDI extension to support QoS- enriched service publication and discovery has generated several research efforts. ShaikhAli's approach [9] is based on the extension of the UDDI business service structure, but potential QoS changes are not considered. Chen et al [10] proposed a registry that receives reports made by consumers to generate QoS summaries for invoked web services. Kalepu et al [11] evaluated the reputation of a service as a function of three factors: ratings made by users, service quality compliance, and the changes of service quality conformance over time. However, these solutions do not take into account the trustworthiness of QoS reports produced by users, which is important to assure the accuracy of the QoS-based web service selection and ranking results. Liu et al [12] suggested an approach for rates services computationally in terms of their quality performance from QoS information provided by monitoring services and users. The authors also employ a simple approach of reputation management by identifying every requester to avoid report flooding. Diego and Maria [13] proposed an extended Web service architecture to support QoS management. The architecture is currently being integrated with Business Process Management (BPM) Technology. The major contributions are: Extending the WS policy framework to specify QoS policies for web services, extending the UDDI information model and API set to refine service discovery and using tModels to define QoS related concepts.

Tian et al [14] explained the WS-QoS architecture that enables QoS-aware service specifications as well as the broker based web service selection model that enables an efficient QoS-aware service selection. Eyhab and Qusay [15] introduced a mechanism that extends the Web Services Repository Builder (WSRB) of Web Services. It also introduced the Web Service Relevancy Function (WsRF) used for measuring the relevancy ranking of a particular Web service based on client's preferences and QoS metrics. Xu et al [16] presented a web service discovery model that contains an extended UDDI to accommodate the QoS information, a reputation management system to build and maintain service reputations and a discovery agent to facilitate service discovery. A service matching, ranking and selection algorithm is also developed. Demian et al [17] explored different types of requester's QoS requirements and a tree model for requester's QoS requirements. It also proposed a QoS broker based web service architecture which facilitates the requester to select a suitable web service based on QoS requirements and preferences. The Web service selection and ranking mechanism uses the QoS broker based architecture [19]. The QoS broker is responsible for the selection and ranking of functionally similar Web services. The Web service selection mechanism [19] ranks the Web services based on prospective levels of satisfaction of requester's QoS constraints and preferences. Serhani [20] presented a web service selection architecture which employs an extended UDDI registry to support service selection based on QoS, but only the certification approach is used to verify QoS and no information is provided about the QoS specification.

3. WS-QOS BROKER ARCHITECTURE

The architecture consists of the basic web service model components like the web service provider, web service consumer and the UDDI registry. In addition, UDDI registry has the capability to store QoS information using tModel data structure and a WS-QoS Broker component. The WS-QoS Broker assists clients in selecting web services based on a set of QoS parameters. The WS-QoS Broker has four components: Service Publisher [17], Verifier and Certifier, Service Selector [17] and Web Service Storage (WSS) [18]. Broker services may be used to facilitate service registry access. The broker performs the interaction with the UDDI. It provides the QoS management operations.

The broker is also a web service. This enables the architecture deployment in restricted and open environments. Figure 1 presents an agent-based architecture with features that overcome the limitations of existing approaches. The service publisher component facilitates the registration, updating and deletion of web service related information. It gets the business specific and performance specific QoS property values of web services from the service providers. The service provider publishes its service functionality to the UDDI registry through the service publisher after certification and verification. For every service or group of services there exists a service publisher that handles all communication with registries, bindings, negotiations, requests and responses for that service. The service consumer can search the UDDI registry for a specific service through the service selector. The main functionality of the service selector component is to select the most suitable web service satisfying requester's QoS constraints and preferences, along with service functionality. The WS-QoS broker performs the verification and certification tasks. QoS verification is the process of validating the correctness of information described in the service interface as well as the described QoS parameters. The verification will be used as input for the certification process that will be issued when the verification succeed.

The QoS property values obtained from the service providers are verified and certified by the Verifier and Certifier component before registering them into the UDDI registry. The Verifier and Certifier component is implemented within the WS-QoS Broker and is responsible for certifying web services and their provided QoS. A certificate is sent to the web services provider and a copy is stored in the WSS for future use. The web service consumer can verify the advertised QoS with the service selector before binding to a web service. The QoS information is represented in UDDI registry by a tModel, which allows specification, standardization and reuse of QoS related concepts. This extension allows the use of brokers to facilitate service selection according to functional and non-functional requirements, and monitors to verify QoS attributes. QoS represents the non-functional aspects of the service being provided to the web service users. The following QoS parameters are considered:

- **Price**: The cost involved in requesting the service which can be estimated by operation or volume of data
- **Response Time**: Time taken by a service to respond to the client request

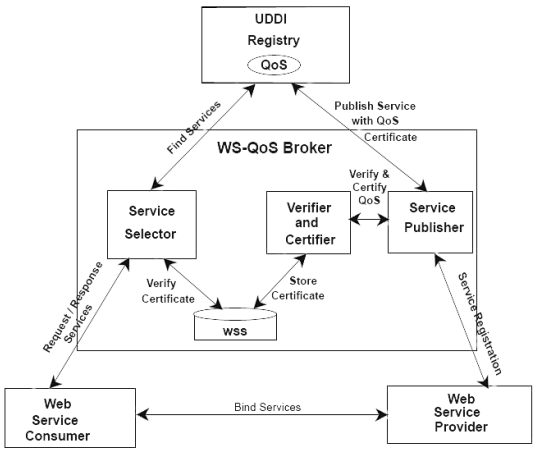

Figure 1. Architecture for WS-QoS Broker

- **Availability**: Percentage of time that the service is operating
- **Throughput**: The maximum requests that can be handled at a given unit in time.

A tModel consists of a key, a name, an optional description and a Uniform Resource Locator (URL), that point to the location where the details about the actual concept can be found. When a service is published in the UDDI registry, a tModel is created to represent the QoS information of the service. It is registered with the UDDI registry and referenced in the bindingTemplate that represents the deployment information of the web service. In the tModel, each QoS metrics is represented as a KeyedReference, which contains the name of a QoS attribute as keyName and keyValue, which contains the value. A service provider should regularly update the QoS information of the services, it publishes, to ensure that the information is accurate and up-to-date. To update the QoS information of a service, the service provider searches the UDDI registry through the service publisher to find the corresponding tModel. It then updates the QoS information in the tModel and saves it back using the same tModel key that was assigned to the tModel when it was created.

The units of QoS attributes are not represented in the *tModel*. We assume default units are used for the values of QoS attributes in the *tModel*. For example, the default unit used for price is CAN$ per transaction, for response time is second, for availability is percentage, and for throughput is transaction per second. For example, a company publishes its Stock Quote service in a UDDI registry with the QoS information.

```
<tModel tModelKey = "somecompany.com: Stock
QuoteService:PrimaryBinding:QoSInformation">
 <name>QoS Information for Stock Quote Service</name>
  <overviewDoc>
    <overviewURL>
      http://<URL describing schema of QoS attributes>
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
        tModelKey="uddi:uddi.org:QoS:Price"
        keyName="Price "
```

```
        keyValue=" 0.01" />
    <keyedReference
        tModelKey="uddi:uddi.org:QoS:ResponseTime"
        keyName="ResponseTime"
        keyValue="0.05" />
    <keyedReference
        tModelKey="uddi:uddi.org:QoS:Availability"
        keyName="Availability"
        keyValue="99.99" />
    <keyedReference
        tModelKey="uddi:uddi.org:QoS:Throughput"
        keyName=" Throughput"
        keyValue="500" />
  </categoryBag>
</tModel>
```

Figure 2. *tModel* with QoS Information

Above given is an example of the QoS Information *tModel*, which contains a *categoryBag*, which is a list of name-value pairs specifying QoS metrics. This *tModel* contains a *categoryBag* that specifies four QoS metrics of Response Time, Throughput, Price and Availability. The *tModelKey* in each *keyedReference* is used as a namespace which provides a uniform naming scheme. The company creates and registers a *tModel* that contains the QoS information for this service before it publishes the service with the UDDI registry.

A typical usage scenario is described here by considering an example in which a requester consumes the Web service of a provider.

- Step1: Initially WS-QoS Broker publishes the interface to the UDDI registry.
- Step2: Web service provider finds the broker interface in UDDI registry.

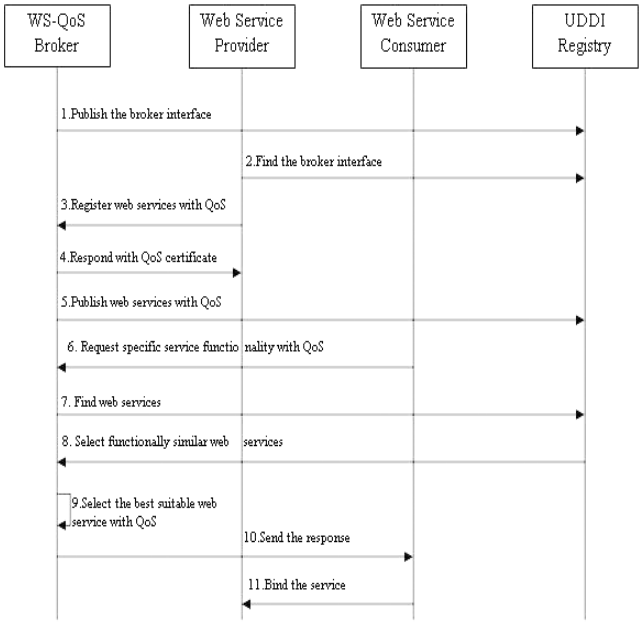

Figure 3. Architectural Component Interactions

- Step3: The service provider registers the web service with the service publisher and provides functional and non-functional information about the offered services.
- Step4: The Verifier and Certifier component in the WS-QoS broker verifies the QoS information and issues a certificate.
- Step5: A copy of the QoS certificate is stored in WSS and a copy is sent to the service provider.
- Step6: The service publisher then publishes the web service in the UDDI registry along with the QoS certificate.
- Step7: The web service consumer requests service selection and provides functional and QoS requirements.
- Step8: The service selector selects a service in the UDDI registry according to the required service functionality and QoS requirements of the application.
- Step9: Service selector can verify the provided QoS certificate with the one stored in the WSS.
- Step10: The service selector then reports the selected service back to the application.
- Step11: The web service consumer then binds the web service from the service provider.

3.1 Service Publisher

The service publisher component communicates with the service provider and the UDDI registry. The web service provider registers the business and web service related information with the service publisher. It also gets the specific QoS property values of web services from providers. Once the QoS property values and other information are obtained from the provider it is hand over to the Verifier and Certifier component. The QoS information is verified and certified before publishing it in the UDDI registry.

3.2 Verifier and Certifier

This is the key component of the WS-QoS Broker that performs the verification of the QoS information supplied by the service provider and issues a certificate to the service provider through the service publisher. This QoS certificate assures that the QoS offered by the provider conform to their descriptions. The service provider initiates the verification process through the service publisher by supplying the QoS property values. The verifier is provided with the WSDL document and additional information about resources available at the provider's platform. The verifier performs the testing of the service URI, the XML schema definition, the service binding information and the availability of all operations described in the service interface. Verifier also performs the verification of the QoS information introduced in the service interface.

The QoS verification is conducted through a set of test cases generated by the verifier. For each test, additional information like server capacity, network bandwidth about the provider and its web services are needed. The four QoS parameters (Response Time, Availability, Throughput, and Price) are also verified. The verification process is done in three levels: General web services information verification, WSDL content verification and QoS verification. A web service is said to be compliant with a given level when it passes the corresponding tests described in the verification document. Based on this, web services can be classified into three classes. Class A includes web services for

which all verification tests have succeeded. Class B includes web services for which more than 80% of the verification tests have succeeded. Class C contains the services for which most of the verification scenarios have failed.

Once the verification process is completed successfully, the certification process is initiated. The certifier issues a certificate to the service provider through the service publisher which indicates that the offered QoS conform to their descriptions. The main responsibility of the certifier is to certify the web services and their provided QoS. A copy of the certificate sent to the service provider, which is also stored in the WSS for future use. The certificate includes information such as certificate number, certificate issue date, number of years in business and service location. In case, if the certificate cannot be issued, feedback will be sent to the provider. After the QoS certification process, the service publisher can register the functional description of the web service and the certified QoS information with the UDDI registry.

3.3 Service Selector

The service selector component is concerned with selecting the most suitable web service satisfying the consumer's QoS constraints and the specific service functionality. It receives messages from the web service consumer, specifying the service functionality along with the QoS constraints. Based on the received requirements specification, it discovers functionally similar web services from the UDDI registry. The service selector can check the validity of the QoS information in the UDDI registry by comparing the QoS certificate provided by the Verifier and Certifier with the one stored in the WSS.

3.4 QoS Matching, Ranking and Selection Algorithm

A web service consumer sends a service selection request to the service selector, which then contacts the UDDI registry to find services that meet the customer's functional and QoS requirements. A service is said to be a "match" if it satisfies the customer's functional requirements and its QoS constraints. If no matched service is found by the matching process, the service selector returns an empty result to the customer. If multiple services match the functional and QoS requirements, the service selector calculates a QoS score for each matched service based on the dominant QoS attribute specified by the customer, or on the default dominant attribute, average response time. The best service is assigned a score of 1, and the other services are assigned scores based on the value of the dominant QoS attribute. The top M services (M being the maximum number of services to be returned as specified by the customer) with the highest QoS scores are returned to the customer. If M is not specified, one service is randomly selected from those services whose QoS score is greater than LowLimit. Figure 4 shows the details of QoS matching algorithm. It is comprised of the following methods: **getServiceQoS** finds the QoS advertisements of a service in a UDDI registry. **qosMatchAdvert** finds if the QoS advertisements of a service satisfies the QoS requirements.

```
qos Match(services,qosReqt){
  matches=Service[];
  for each s in services
//get Qos info from UDDI
qosAds=getServiceQoS(s);
//if QoS info available and satisfies QoS requirements
If(qosAds!=null&& qosMatchAdvert(qosAds,qosReqt)
matches.add(s);
  end for
  return matches;
}
```

Figure 4. QoS Matching Algorithm

Figure 5 shows the details of QoS ranking algorithm. It consists of the following methods: **calculateQoSScore** calculates QoS scores for the services that meet the QoS requirements. **sortByQoSScore** returns a list of services sorted by the QoS score in descending order.

```
//rank matches with QoS information
qosRank(services,qosReqt){
  //calculate QoS scores
  services= calculateQoSScore(services,qosReqt);
  //sort the result by QoS score in descending order
  services= sortByQoSScore(services);
  return services;
}
```

Figure 5. QoS Ranking Algorithm

Figure 6 shows the details of the service selection algorithm. If the maxNumServices, that is the maximum number of services to be returned by service selector, is greater than 1, then the top maxNumServices services are returned if the option is "random", or the top maxNumServices services with the highest QoS or overall scores if the option is "byQoS" or "byOverall", are returned to the customer. Otherwise, one service is randomly selected if the option is "random", or from those whose QoS or overall score is greater than LowLimit if the option is "byQoS" or "byOverall".

```
// select services according the max number of services to be
  // returned
selectServices (matches, maxNumServices, option) {
  selection = service [];
  if maxNumServices > 1
    i = 0;
    while i < maxNumServices && i < matches.size()
      selection.add(matches[i]);
      i++;
else
     candidate = service [];
     if option  == "random"
        candidate = matches;
else
   for each s in matches
      if option == "byQoS"
        if s.QoSScore >= LowLimit
          candidate.add(s);
     else
        if s.overallScore >= LowLimit
          candidate.add(s);
   end for
  pickNum = random (0, candidate.size() );
  selection.add(candidate[pickNum]);
return selection;     }
```

Figure 6. The service selection algorithm

4. CONCLUSION

The integration of the various QoS properties is essential for the success of the web service technology. Due to the increasing popularity of Web services technology and the potential of dynamic service selection and integration, multiple service providers are now providing similar services. QoS is a decisive factor to distinguish functionally similar Web services. The major problem with the current web service selection is the absence of a mechanism that considers QoS properties for the web service selection. We propose an approach that reduces the complexity of matching user requests according to the specified functional and QoS requirements. We implement a new WS-QoS broker based architecture that solves the problems associated with web service selection. The broker performs the process of publishing and selection of web services. Our suggested theoretical architecture will be based and implemented on QoS properties. An amount of services is needed to test the performance of the system. This will enable a more flexible, and trustable architecture. Results of this work will be reported in a future paper. Future work involves enhancing the capabilities of the proposed architecture to handle other QoS attributes and adapting the architecture to support mobile Web services.

5. REFERENCES

[1] Martin-Diaz, O., Ruiz-Cortes, A., Corchuelo, R., and Toro, M., 2003, "A Framework for Classifying and Comparing Web Services Procurement Platforms", Proc.of 1st Int'l Web Services Quality Workshop, Italy, pp. 37-46.

[2] Blum, A., 2004, "UDDI as an Extended Web Services Registry: Versioning, quality of service, and more". White paper, SOA World magazine, Vol. 4(6).

[3] Latimer-Livingston, N.S., Graham, C., Correia, J.M. and Schroder, N., 2003, "Survey Shows Why Firms Undertake Web Services Projects", Technical report, Gartner Group.

[4] Baroudi, C. and Halper, F., 2006, "Executive Survey: SOA Implementation Satisfaction", Technical report, Hurwitz and Associates.

[5] Ran, S., 2004, "A Model for Web Services Discovery with QoS". ACM SIGEcom Exchanges, Vol. 4(1), pp.1–10.

[6] Majitha, S., Shaikhali, A., Rana, O. and Walker, D., 2004, "Reputation based semantic service Discovery", In Proc. Of the 13 th IEEE Intl Workshops on Enabling Technologies Infrastructures for collaborative Enterprises (WETICE), Modena, Italy, pp.297-302.

[7] Rajendran, T. and Balasubramanie, P., 2009, "An Efficient Framework for Agent-Based Quality Driven Web Services Discovery", IEEE International conference on Intelligent Agent and Multi Agent Systems (IAMA2009), Chennai.

[8] Keller, A. and Ludwing. H., 2002, "The WSLA framework: Specifying and Monitoring Service Level Agreements for Web Services", IBM Research Report.

[9] ShaikAli, A., Rana, O.F., Al-Ali, R., and Walker, D.W., 2003, "UDDIe: An extended registry for web services". In Proc. Of the Symposium on Applications and the Internet workshops, IEEE CS, pp 85-89.

[10] Chen, Z., Liang-Tien, C., Silverajan, B. and Bu-Sung, L., 2003, "UX-An architecture providing QoS-aware and federated support for UDDI". In proc. of the Int'l Conf. on web services, CSREA Press, pp 171-176.

[11] Kalepu, S., Krishnaswamy, S. and Loke, S.W., 2004, "Reputation = f (User Ranking, Compliance, Verity)", Proceedings of ICWS'04.

[12] Liu, Y., Ngu, A. and Zheng, L., 2004, "QoS Computation and Policing in Dynamic Web Service Selection", Proceedings of WWW 2004 Conf.

[13] Diego Zuquim Guimaraes Garcia and Maria Beatriz Felgar de Toledo, 2006, "A web service Architecture providing QoS Management", Institute of Computing, University of Campinas, Sao Paulo, Brazil, pp -189-198.

[14] Tian, M., Gramm, A., Ritter, H. and Schiller, J., 2004, "Efficient Selection and Monitoring of QoS aware Web Services with the WS-QoS Framework". Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04) Exchanges, vol. 4, no. 1, pp. 1–10.

[15] Eyhab Al- Masri, and Qusay H. Mahmoud, 2007, "QoS-based Discovery and Ranking of Web services", Proceedings of IEEE International Conference.

[16] Ziqiang Xu, Patrick Martin, Wendy Powley and Farhana Zulkernine, 2007, "Reputation Enhanced QoS-based Web services Discovery", IEEE International Conference on Web Services (ICWS 2007).

[17] Demian Antony D' Mello, V.S.Ananthanarayana and Santhi.T, 2008, "A QoS Broker Based Architecture for Web Service Selection", Proceedings of IEEE International Conference.

[18] Rajendran, T. and Balasubramanie, P., 2009, "An Agent-Based Dynamic Web Service Discovery Framework with QoS Support", International J. of Engg. Research & Indu. Appls. (IJERIA), Vol.2(5),pp 1-13.

[19] Demian. A. D'Mello and Ananthanarayana, V.S., 2008, "A QoS Model and Selection Mechanism for QoS-Aware Web Services", Proceedings of the International Conference on Data Management (ICDM 2008).

[20] Serhani, M.A., Dssouli, R., Hafid, A. and Sahraoui, H., 2005, "A QoS broker based architecture for efficient Web services selection". In Proc. of the IEEE Int'l Conf. on Web Services, IEEE CS, pages 113–120.