A QoS Broker Based Architecture for Dynamic Web Service Selection

Demian Antony D'Mello, V.S. Ananthanarayana and Santhi T.
Department of Information Technology, National Institute of Technology Karnataka, India
{demian,anvs,santhi}@nitk.ac.in

Abstract

The increasing number of Web services over the Web makes the requester to use tools to search for suitable Web services available throughout the globe. UDDI is the first step towards meeting these demands. However the requester's demand may include not only functional aspects of Web services but also non-functional aspects like Quality of Service (QoS). There is a need to select the most suitable (qualitatively optimal) Web service based on the requester's QoS requirements and preferences. In this paper we explore the different types of requester's QoS requirements (demands) with illustrations. We propose the QoS broker based architecture for dynamic Web service selection which facilitates the requester to specify his/her QoS requirements along with functional requirements. The paper presents the Web service selection mechanism which selects the best (most suitable) Web service based on the requester's functional and quality requirements.

1. Introduction

The Web service technology is becoming popular because of its potential in many areas. A Web service is an interface that describes a collection of operations that are network accessible through standardized XML messaging [1]. Web services can be advertised, located and used across the internet using a set of standards such as SOAP, WSDL and UDDI. The Web service fulfills a specific task or a set of tasks and it can be used alone or with other Web services to carry out a complex aggregation or a business transaction [1]. The present Web service architecture is based upon the interactions between three roles: service provider, service registry and service requester. The interactions among them involve publish, find and bind operations [1]. The heavily increasing number of Web service providers on the Web supporting numerous Web services having same or similar functionality made a way for the consumers to use tools and techniques to search for suitable Web services based on their requirements. Some attempts have been made concerning the discovery of Web services based on their non-functional (what they serve) [1,3] and functional properties (how they serve) [4,5]. In Web service discovery mechanism the matchmaking is explored through many ways such as keyword and category based [1,3], behavioral signature i.e. operational level description based [4], domain specific description based [6], interface signature based [5] and semantic description i.e. input, output, precondition, effect (IOPE) based [7,8] approaches. The limitation of Web service discovery mechanism is that it returns multiple Web services having similar or same functionality with no distinction. The Web service selection is the process of choosing a suitable Web service from functionally similar Web services. In literature the Web service selection is made based on personalization [9], requester's trust and connection policy [10,11], requester's past experience with the Web service [12] and the QoS [13,2,14,22].

Quality of Service in Web services is a combination of several qualities of a Web service and it is a measure for how well the Web service serves the requester. The efforts are on to define QoS models and its impact on Web service architecture [15,16,24]. The paper [16] describes several *performance specific* QoS properties like Availability, Security, Response Time, Accessibility, Reliability, Integrity and Throughput. The *requester's response specific* QoS properties like Reputation [24], Compliance [23], and Successability are estimated based on requester's feedback which is obtained after service consumption. The *business specific* (price specific) QoS properties like Execution Cost/Price, Penalty Rate (Penalty), and Compensation Rate (Compensation) are defined in [17] which are obtained from Web service providers during service registration. The important feature of all QoS properties is that *from requester's point of view they are either increasing or decreasing in nature*. For an increasing QoS property a higher value indicates the better Web service quality and vice versa. For example

QoS property Response Time is decreasing since lesser value of Response Time indicates the better Web service.

QoS can be used to select and rank the Web services by extending standard service oriented architecture (SOA) [18,19]. In this architecture, the Web service is selected by matching requested QoS property values against the potential Web service QoS property values. In literature, the Web service is selected by taking the requester's average preference for QoS properties [19]. A single QoS property (e.g. Price) is also used to filter and rank the functionally similar Web services [20]. The Web services are also ranked based on multiple QoS properties by normalizing the QoS property values to a non-negative real valued number ranging from zero to one [2,19,13,14]. The normalized values are then multiplied with the weights representing requester's preferences. Finally, the multiplied values are added to get a score which is used to discriminate the Web services. The Web services are also ranked by computing correlation (Euclidian distance) between requested QoS property values and the potential Web service QoS property values [14]. In literature, so far no work has been done towards the selection and ranking of Web services based on requester's multiple requirements on QoS with varied preferences. In this paper we explore different types of requester's QoS requirements and a tree model for requester's QoS requirements. The paper also proposes QoS broker based Web service architecture which facilitates the requester to select a suitable Web service based on his/her QoS requirements and preferences.

The rest of the paper is organized as follows. Section 2 addresses requester's QoS requirements and modelling. Section 3 describes QoS broker based architecture for dynamic Web service selection. In section 4 the paper presents Web service selection mechanism with an illustration. Section 5 presents the implementation details. Section 6 draws conclusions.

2. Requester's QoS Constraint Modelling

The Web service requester normally expects some requirements on QoS are to be satisfied by the Web services. We call these requirements on QoS properties as QoS constraints. The QoS constraint is a relational expression defined on some QoS properties. For example, in shopping domain the buyer may have QoS constraints like "delivery should be within 4 days", "delivery price should not exceed $5", Web service with reputation greater than 6 (out of ten) etc. The QoS constraints will be different for individual requesters. Normally the buyer looks for an inexpensive seller service that supports quick delivery but some buyers focus either on quick delivery or lesser delivery price. Thus Web service requester can enforce any number of QoS constraints on multiple QoS properties with varied preferences to select the desired Web service.

2.1 QoS Constraint Types

We categorize QoS constraints based on QoS constraint structure as *simple* and *composite* QoS constraints. A simple QoS constraint normally deals with one QoS property. A simple QoS constraint normally takes the format Q_i *cp* V_i where Q_i refers to QoS property, *cp* refers to comparison operator (<, >, ≤, = and ≥) or membership operator (*in*) and V_i refers to expected single value or range of values for Q_i. We further classify simple QoS constraint as *point, implicit range* and *explicit range* QoS constraints based on the nature of *cp* and V_i. A simple QoS constraint with equality operator (=) is called as *point* QoS constraint. For example buyer might say "I need a seller who delivers an item in one day". This is point constraint and can be written as "time = 1". A simple QoS constraint with comparison operators <, >, ≤ and ≥ is referred as *implicit range* QoS constraint. In implicit range QoS constraint either lower bound or upper bound of V_i is implicit i.e. it refers to the minimum or maximum value of Q_i. For example buyer might say "I need a seller having reputation (out of ten) more than 7". This is implicit range QoS constraint (reputation is expressed in terms of integer value within the range 1-10) which can be written as "reputation > 7". A simple QoS constraint with explicit lower and upper bound values for V_i is referred as *explicit range* QoS constraint. For example buyer might say "I need a seller whose delivery price is between 4 to 8 dollars". This is explicit range QoS constraint which can be written as "price *in* [4-8]".

Composite QoS constraint is composed of multiple simple QoS constraints using constraint composition operators *AND* and *OR*. For example buyer might say "I am interested in a seller with price less than $3 and delivery time less than 3 days". This is composite QoS constraint which can be represented as "price < 3 *AND* delivery time < 3". A composite QoS constraint takes the form C_1 *op* C_2 *op* C_3 *op* ...*op* C_P where C_i refers to simple QoS constraint and *op* denotes constraint composition operator *AND* or *OR*. In general a QoS constraint takes the form $(Q_i \; cp \; V_i) \; (op \; (Q_j \; cp \; V_j))^*$. The requester can enforce either simple or composite QoS constraint to choose a good quality Web service.

2.2 QoS Constraint Modelling

Consider requester's QoS constraint on several QoS properties with preference to simple and composite QoS constraints. We model requester's QoS constraints as follows:

AND-OR Tree. An AND-OR tree is a non-empty rooted tree of order *N,* with finite number of nodes and each node can contain *zero* or *two* or *N* ($N>2$) child nodes. A node with no child is called as a *leaf* and the node with *C* ($2 \leq C \leq N$) child nodes is referred as *internal* node. The internal node contains one item of information i.e. *AND* or *OR* and leaf node contains finite items of information.

We assign level-0 for all nodes having no child and the level of internal node can be computed as the maximum among the levels of its child nodes + 1. Let *H* be the height of an AND-OR tree, then the level of root node is *H* and the levels of internal nodes will be in between 1 and *H*-1.

Weighted AND-OR Tree. A Weighted AND-OR tree is an AND-OR tree where every edge between parent and child node is labeled with a non-negative real number in an interval (0, 1) such that for any parent node the sum of edge labels (weights) of all child nodes is equal to one i.e. for any parent node *P* with *C* ($2 \leq C \leq N$) child nodes, the sum of edge weights W_{PCi} ($1 \leq i \leq C$) is equal to 1.

Quality Constraint Tree (QC tree). A Quality Constraint tree is a Weighted AND-OR tree whose leaf node contains either three or four information items. A leaf contains QoS property (Q_i), comparison or membership operator (*cp*) and expected QoS property value (V_i) for Q_i. The internal node refers to constraint composition operator *op*. The label W_{XY} on the edge between any two nodes *X* and *Y* represents the preference for sub-tree rooted at *Y* while traversing from root to leaf i.e. the edge label represents requester's preference to either simple or composite QoS constraint defined for the sub-tree rooted at node *Y*. Thus leaf node represents simple QoS constraint and any sub-tree rooted at internal node represents composite QoS constraint.

The requester's QoS constraint can be represented using QC tree. Consider the buying scenario where the buyer looks for a electronic article seller Web service with the following requirements: (a) Response time (RT) < 4 AND Execution Price (PR) < \$6 AND Penalty (PT)< 30 AND Accessibility (AC) > 0.5 with preference 0.3, 0.4, 0.2 and 0.1 to requested QoS properties (b) Reputation (RP) > 6 AND Response time (RT) < 6 AND Price (PR) < \$9 with preference 0.4, 0.3 and 0.3 to requested QoS properties. Buyer expects a good service that satisfies one of the QoS requirements i.e. either QoS constraint (a) or QoS requirement (b). Figure 1 depicts the QC tree for the buyer's QoS constraints. The leaf nodes are represented using rectangle and internal nodes with circles. The level of the root node is computed as 2.

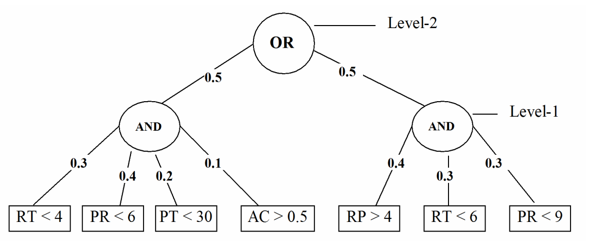

Figure 1. QC Tree for Buyer's QoS Constraints

3. A QoS Broker Based Architecture

We propose QoS broker based architecture for Web services with an objective of selecting the best Web service that satisfies requester's QoS constraints and preferences. Figure 2 depicts the different roles and operations supported by the QoS broker based Web service architecture. The architecture assumes that the Web service requesters and the providers will use the QoS vocabulary defined as in [17] for the purpose of Web service selection and QoS registration.

3.1 Architecture Roles and Operations

We define two additional roles to the conceptual Web service architecture [1] called *QoS broker* and *QoS Registry*. The architecture also defines additional operations which are listed in Figure 2. The QoS broker is defined between Web service registry and the requester which facilitates the requester to specify his/her QoS constraints and preferences to select the desired Web service. The select operation is defined between QoS broker and the requester to select the best Web service based on requester's QoS constraints and preferences. From an architectural perspective, the QoS broker is a middleware which can be implemented as a Web service. The QoS broker registers and edits (if required) the Web service QoS property values through the QoS registry. The QoS registry provides the facilities to search price, performance and requester's response sensitive QoS information of Web services.

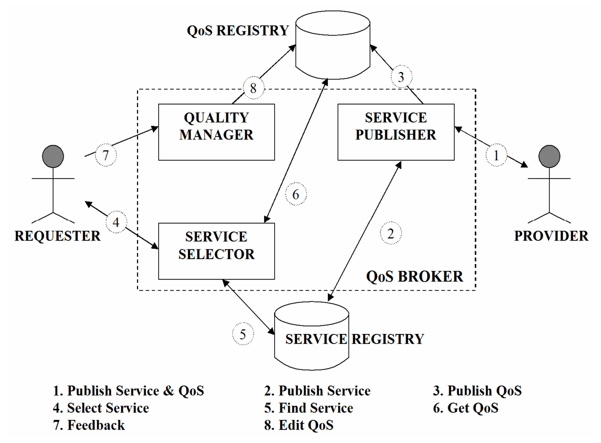

Figure 2. A QoS Broker Based Architecture

3.2 Component Interactions

We design the QoS broker with three functional components: *Service Selector*, *Service Publisher* and *Quality Manager*. The structure and interactions among various components and architectural roles are depicted in Figure 2. For each component, we define a set of functions to fulfill the requester's objective of selecting the best Web service that satisfies his/her QoS constraints and preferences.

Service Publisher. The Service publisher component facilitates the registration, updating and deletion of business and Web service related information. In addition to this, it gets the business specific and performance specific QoS property values of Web services from the providers. The obtained QoS property values are finally verified and certified [22] by the QoS broker before registering them into QoS registry. The service selector also allows modification and deletion of QoS property values.

Service Selector. The main functionality of this component is to select the most suitable Web service satisfying requester's QoS constraints and preferences. The responsibilities of this component are:

(a) To receive messages containing the requested service functionality along with QoS constraints from the requester.
(b) To discover functionally similar Web services from the Web service registry through functionality matching [1,3,5].
(c) To select the most suitable Web service based on the requester's QoS constraints and preferences.

Quality Manager. The main objective of this component is to estimate the requester's response specific QoS property values like Reputation, Successability and Compliance through requester's response (feedback) on service execution. The feedback from the requester is obtained under the assumption that, the requesters are willing to return the QoS property values to the QoS broker when asked by it after the service consumption and the received QoS values from the requester can be trusted.

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

- Step 1. Initially, the Web service provider registers the Web service and its business, performance specific QoS with the service publisher component of QoS broker.
- Step 2. Service publisher component publishes service information with the service registry.
- Step 3. The QoS broker certifies the performance specific QoS properties after verification. After certification, the QoS property values are published with the QoS registry.
- Step 4. The requester sends the request to the QoS broker for service selection by providing the functional needs and QoS constraints (QC Tree).
- Step 5. The service selector component discovers functionally similar Web services from the UDDI registry (service registry).
- Step 6. The service selector component now gets the required QoS property values for the discovered Web services from the QoS registry and executes the
- Step 7. After consuming Web service from the provider, the requester will send the feedback about the service execution to the quality manager.
- Step 8. Based on requester's feedback, the quality manager component edits the QoS properties of specific Web service.

4. The Web service Selection Mechanism

The selection mechanism takes QC tree and candidate Web services (functionally similar Web services returned by Web service registry through functionality matching) as an input and results in the best Web service that meets requester's QoS constraints. The mechanism traverses QC tree in level order fashion (level-0 to level-H) and treats both leaf and internal nodes in a different manner. At leaf node, the selection mechanism performs the following *four* actions: (0) *Preprocessing* (1) *Filtering* (2) *Scaling* and (3) *Ranking*. The optional preprocessing phase is necessary if the leaf node contains the QoS properties like Penalty, Compensation, Reputation and Compliance. The Compensation and Penalty are expressed in terms of percentage of execution price. Thus subsequent processing (filtering) should be based on actual monetary value involved in the QoS constraint. This phase is essential for the simple QoS constraints involving QoS properties like Reputation [24] and Compliance [23] since these QoS properties are computed as a function of other QoS properties. In filtering phase the Web services that satisfy simple QoS constraint defined at the leaf node are selected. The scaling phase normalizes the QoS values of selected Web services to a non-negative real valued number in an interval [0,1] using min-max normalization technique [14]; where the higher normalized values represent higher level of quality. In ranking phase normalized values are multiplied with the weight (preference given to the QoS constraint) to get the new scores for the Web services.

At the internal node, the selection procedure performs *two* actions namely *Filtering* & *Ranking* which are dependent on the type of node (*AND*/*OR*). In filtering phase if the node is *AND* then the Web service present in *all* its child nodes is selected. If the node is *OR* then *distinct* Web services in the descending order of their scores are selected from its child nodes. In ranking phase if the node is *AND* then the score of selected Web service is computed as the sum of scores of selected Web service at its child nodes multiplied with the weight of sub-tree rooted at *AND* node. If the node is *OR* then the score of selected Web service is multiplied with the weight of sub-tree rooted at *OR* node. After ranking Web services at the root node, Web services are sorted in the descending order of their score. Now the Web services are found in the order of prospective levels of satisfaction of requester's QoS constraints and the service selector component returns the first Web service as the most suitable Web service to the requester If the selection mechanism returns empty Web service list to the requester for a given QoS constraints then requester has to revise the enforced QoS constraints to get the best available Web service.

The algorithm also optimizes the processing while handling *AND* nodes. While encountering *AND* nodes if the algorithm finds a child with no selected Web services then it attaches Nil to *AND* node and the algorithm skips the evaluation of other child nodes. As an illustration for the selection mechanism, consider the QC tree of Figure 1. Assume that, the Web service registry returns 5 Web services (WS_1 to WS_5) for the electronic item seller Web service request to the QoS broker. The service selector component of the QoS broker retrieves the required QoS property values for the candidate Web services from the QoS registry. The QoS property values for 5 Web services are given in the order of Response Time (RT), Price, Penalty, Accessibility and Reputation (Rep) as follows: $QoS(WS_1)=\{2, 10, 20, 0.3, 4\}$, $QoS(WS_2)=\{5, 3, 30, 0.4, 8\}$, $QoS(WS_3)=\{3, 5, 10, 0.5, 7\}$, $QoS(WS_4)=\{7, 8, 35, 0.8, 3\}$ and $QoS(WS_5)= \{9, 7, 20, 0.2, 4\}$. Figure 3 shows the trace of Web service selection algorithm with the Web services selected at each node and their computed values. The Web service WS_3 is selected at left AND node and Web services WS_2, WS_3 are selected for right AND node as these Web services are found in all the child nodes of AND node. At root node, only two Web services are selected. Finally the selection mechanism recommends Web service WS_2 to a requester as best Web service.

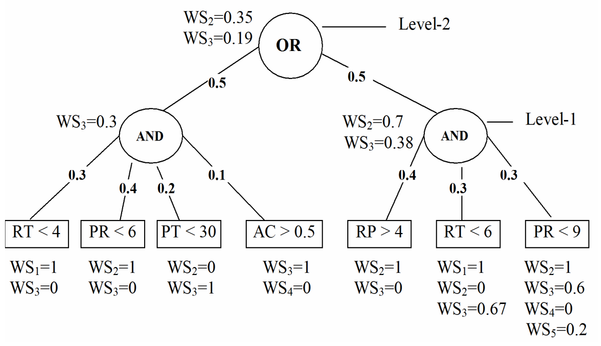

Figure 3. The Trace of Selection Mechanism

5. Implementation and Experiments

The QoS broker is implemented on Windows XP platform using Microsoft Visual Studio .NET development environment and Microsoft visual C# as a programming language. To enable the interaction between .NET program and the UDDI-compliant server, we have used Microsoft UDDI .NET 2.0 Beta 1 SDK. We have used SAP UDDI V3 Test Public Business Registry [21] enquiry API (http://uddi.sap.com/uddi/api/inquiry) to retrieve the candidate Web services from SAP UDDI test public registry. We implemented the QoS registry as a Web service with cost sensitive; a performance sensitive search options. The QoS registry reads QoS values of Web services form the Microsoft SQL Server 2000 database. The six QoS properties for ten Web services are considered for the experimentation.

The QoS broker reads the functionality description (keyword) of a service and the requester's QoS constraints (by taking N= 4) for the selection. The simple QoS constraints are supplied to the QoS broker in the following format: {Constraint number, Comparison Operator, QoS property, Value, Weight} and composite QoS constraints are fed in the following format: {Constraint number, Composition Operator, C_1, C_2, C_3, C_4, Weight}; where C_1, C_2, C_3 and C_4 are simple QoS constraint references (Constraint numbers). After the execution of selection mechanism, the QoS broker returns the best Web service to the requester.

6. Conclusion

Quality of Service (QoS) is a decisive factor to distinguish functionally similar Web services. In this paper we introduced the QoS broker based Web service architecture which selects the best Web service for the requester based on his/her QoS constraints. The paper explores the different types of requester's QoS constraints and suggests a Web service selection mechanism which is defined on the tree model of QoS constraint. We implemented the QoS broker and QoS registry for the purpose of experimentation. The experimentation results proved the importance of QoS in distinguishing the functionally similar Web services.

7. References

1. H. Kreger: Web Services Conceptual Architecture (WSCA 1.0), Published in May 2001, www.ibm.com/software/solutions/webservices/pdf/wsca.pdf.
2. Y. Liu, A. H. H. Ngu, and L. Zeng: QoS Computation and Policing in Dynamic Web Service Selection, Proceedings of the WWW 2004, ACM 1-58113-912-8/04/0005, pp. 66-73, 2004.
3. UDDI Technical White Paper, www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, 2000.
4. Z. Shen and J. Su: Web Service Discovery Based on Behavior Signatures, Proceedings of the IEEE International Conference on Services Computing 2005.
5. Y. Wang and E. Stroulia: Flexible Interface Matching for Web-Service Discovery, Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), IEEE 2003.
6. D. Rocco and J. Caverlee: Domain-specific Web Service Discovery with Service Class Descriptions, Proceedings of the IEEE International Conference on Web Services (ICWS'05), IEEE 2005.
7. S. B. Mokhtar, A. Kaul, N. Georgantas, and V. Issarny: Towards Efficient Matching of Semantic Web Service Capabilities, Proceedings of the International Workshop on Web Services Modeling and Testing (WS-Mate 2006).
8. X. Gao, J. Yang, and M. P. Papazoglou: The Capability Matching of Web Services, ISBN 0-7695-1857-5/02, IEEE, 2002
9. W. T, Balke, and M. Wagner: Towards Personalized Selection of Web Services, Proceedings of the WWW 2003, ISBN 963-311-355-5, 2003.
10. S. Ali, S. A. Ludwig, and O. F. Rana: A Cognitive Trust-based Approach for Web Service Discovery and Selection, Proceedings of the Third European Conference on Web Services (ECOWS'05), IEEE 2005.
11. M. Marchi, A Mileo, and A. Provetti: Declarative Policies for Web Service Selection, Proceedings of the IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05), IEEE 2005.
12. Julian Day: Selecting the Best Web Service, http://bistrica.usask.ca/MADMUC/Pubs/day880.pdf.
13. L. H. Vu, M. Hauswirth, and K. Aberer: QoS Based Service Selection and Ranking with Trust and Reputation Management, Proceedings of the CoopIS/DOA/ODBASE 2005, LNCS 3760, pp. 466-483, 2005.
14. L. Taher, R. Basha and H. E. Khatib: Establishing Association between QoS Properties in Service Oriented Architecture, Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05). IEEE 2005.
15. D. A. Menasce: QoS Issues in Web Services, IEEE Internet Computing, pp. 72-75, Nov-Dec 2002.
16. Mani and A. Nagarajan: Understanding quality of service for Web services, www.emeraldinsight.com/Insight/html/Output/Published/EmeraldFullTextArticle/Pdf/1720140505.pdf, Published in 2002.
17. G. Yeom, T. Yun and D. Min: A QoS Model and Testing Mechanism for Quality-driven Web Services Selection, Proceedings of the Second International Workshop on Collaborative Computing, Integration, and Assurance (SES-WCCIA'06), IEEE 2006.
18. M. A. Serhani, R. Dssouli, A. Hafid, and H. Sahraoui: A QoS broker based architecture for efficient web service selection, Proceedings of the IEEE International Conference on Web Services (ICWS'05), IEEE 2005.
19. J. Hu, C. Guo, H. Wang, and P. Zou: Quality Driven Web Services Selection: Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'05), IEEE 2005.
20. M. Kerrigan: Web Service Selection mechanisms in the Web Service Execution Environment (WSMX), Proceedings of the SAC'06, ACM 1-59593-108-2/06/0004, pp. 1664-1668, 2006.
21. SAP UDDI V3 Test Public Business Registry: http://udditest.sap.com/webdynpro/dispatcher/sap.com/tc~uddi~webui~wdp/UDDIWebUI/.
22. Shuping Ran: A Model for Web Services Discovery With QoS, ACM 2003.
23. K. Sravanthi, K. Shonali and S. W. Loke: Reputation= f(User Ranking, Compliance, Verity), Proceedings of the IEEE International Conference on Web Services (ICWS'04), IEEE 2004.
24. Demian Antony D'Mello and V.S. ananthanarayana: A Quality of Service (QoS) Model for Web Services, proceedings of the First International Conference on Emerging Technologies and Applications in Engineering, Technology, and Sciences (ICETAETS 2008), Jan 14-15, 2008, Vol 1, pp. 832-837.