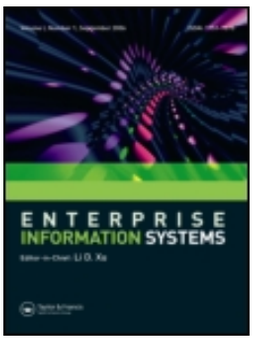

Enterprise Information Systems

ISSN: 1751-7575 (Print) 1751-7583 (Online) Journal homepage: https://www.tandfonline.com/loi/teis20

Dynamic selection mechanism for quality of service aware web services

Demian Antony D'Mello & V. S. Ananthanarayana

To cite this article: Demian Antony D'Mello & V. S. Ananthanarayana (2010) Dynamic selection mechanism for quality of service aware web services, Enterprise Information Systems, 4:1, 23-60, DOI: 10.1080/17517570903159467

To link to this article: https://doi.org/10.1080/17517570903159467

Enterprise Information Systems
Vol. 4, No. 1, February 2010, 23–60

Taylor & Francis
Taylor & Francis Group

Dynamic selection mechanism for quality of service aware web services

Demian Antony D'Mello* and V.S. Ananthanarayana

Department of Information Technology, National Insitute of Technology Karnataka, Srinivas Nagar, Mangalore, 575 025, Karnataka, India

(*Received 25 February 2009; final version received 4 June 2009*)

A web service is an interface of the software component that can be accessed by standard Internet protocols. The web service technology enables an application to application communication and interoperability. The increasing number of web service providers throughout the globe have produced numerous web services providing the same or similar functionality. This necessitates the use of tools and techniques to search the suitable services available over the Web. UDDI (universal description, discovery and integration) is the first initiative to find the suitable web services based on the requester's functional demands. However, the requester's requirements may also include non-functional aspects like quality of service (QoS). In this paper, the authors define a QoS model for QoS aware and business driven web service publishing and selection. The authors propose a QoS requirement format for the requesters, to specify their complex demands on QoS for the web service selection. The authors define a tree structure called quality constraint tree (QCT) to represent the requester's variety of requirements on QoS properties having varied preferences. The paper proposes a QoS broker based architecture for web service selection, which facilitates the requesters to specify their QoS requirements to select qualitatively optimal web service. A web service selection algorithm is presented, which ranks the functionally similar web services based on the degree of satisfaction of the requester's QoS requirements and preferences. The paper defines web service provider qualities to distinguish qualitatively competitive web services. The paper also presents the modelling and selection mechanism for the requester's alternative constraints defined on the QoS. The authors implement the QoS broker based system to prove the correctness of the proposed web service selection mechanism.

Keywords: web service selection; quality of service; QoS constraint; quality constraint tree; QoS broker; QoS model; alternative constraints; provider qualities; QoS registry; extended quality constraint tree

1. Introduction

Web service technology promises to facilitate the efficient execution and coordination of B2B and B2C e-commerce by integrating business applications over the Internet. A web service is an interface that describes a collection of operations that are network accessible through standardised XML messaging (Kreger 2001). Web services can be advertised, located and used across the Internet using a set of

*Corresponding author. Email: demian.antony@gmail.com

ISSN 1751-7575 print/ISSN 1751-7583 online

DOI: 10.1080/17517570903159467
http://www.informaworld.com

standards such as SOAP, WSDL and UDDI. A web service fulfils 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 (Kreger 2001). The widespread adoption of this technology will enable interoperability between heterogeneous platforms and help the business organisations to solve integration problems of their applications.

The present web service architecture is based on the interactions between three roles: service provider, service registry and service requester. The interactions among them involve publish, find and bind operations (Kreger 2001). The heavily increasing number of web service providers on the Web supporting numerous web services having the same or similar functionalities has made a way for the consumers to use tools to search and select suitable web services based on their requirements. Attempts have been made towards the discovery of web services based on their functional (what they serve) properties (Kreger 2001, Bellwood *et al*. 2002) and non-functional (how they serve) properties (Wang and Stroulia 2003, Makris *et al*. 2006). In web service discovery mechanism, the matchmaking is explored through many ways such as keyword and category based (Bellwood *et al*. 2002), behavioural signature, i.e. operational level description based (Shen and Su 2005), domain specific description based (Rocco and Caverlee 2005), interface signature based (Wang and Stroulia 2003), semantic description based (Chung *et al*. 2005, Wu and Wu 2005, Martin *et al*. 2005) and IOPE (input, output, pre-condition, effect) based approaches (Jaeger *et al*. 2005, Zhuang and YuanFie 2006). The weakness of these web service discovery mechanisms is that they return multiple web services having the similar or same functionality with no distinction. This enforces the requester to choose the required web service by analysing descriptions of the discovered web services. The web service selection is the process of choosing one web service from functionally similar web services for the binding (execution). The published, i.e. classical, web service selection techniques are based on the personalisation (Blake and Wagner 2003), requester's trust policy (Ali *et al*. 2005), connection policy (Marchi *et al*. 2005), requester's past experience with the web service execution (Day and Ralph 2004) and web service quality (Ran 2003, Makris *et al*. 2006).

Quality of service (QoS) in web services is a combination of several qualities of web services and it is a measure of how well a web service serves the requester. QoS awareness in web services is advantageous for both requesters and the providers. For the requester, QoS has an impact on the web service selection and composition. On the other hand, QoS can give web service providers a significant competitive advantage in the e-business domain. The efforts are on to define QoS models and its impact on the web service architecture (Menasce 2002, Yeom *et al*. 2006, Demian and Ananthanarayana 2008a). QoS can be used to select and rank the functionally similar web services by extending standard service oriented architecture (Serhani *et al*. 2005, Hu *et al*. 2005, Maximilien and Singh 2003, Liu *et al*. 2004, Kerrigan 2006, Taher *et al*. 2005). The classical QoS selection approaches for web services adopt matchmaking mechanism defined between the requested QoS and web service's QoS, i.e. the matchmaking mechanism maps the requester's expected QoS with published QoS information of the candidate web services.

In this paper, the authors propose the QoS broker based web service selection mechanism which ranks the functionally similar web services based on the degree of satisfaction of the requester's QoS requirements defined on the multiple QoS properties with varied preferences. The authors design the QoS broker for web

services to facilitate dynamic web service selection, publishing and monitoring of QoS properties of web services.

The remaining part of the paper is organised as follows. Section 2 explains the motivating example and the contributions of this paper. Section 3 presents the QoS model for web services. Section 4 explores different types of requester's QoS requirements and modelling of QoS requirements. In Section 5, the paper presents the QoS broker based architecture for web services. Section 6 explores the web service selection algorithm with an illustration. In Section 7, the authors present the mechanism to distinguish the qualitatively competitive web services. Section 8 presents the requester's alternative QoS constraints for the successful web service selection. In Section 9, the paper presents the analytical experimentation and implementation details of the QoS broker based architecture. In Section 10, the authors review the literature and highlight the novel aspects of this paper. Section 11 draws the conclusion and future work.

2. Motivation and contribution

The authors begin by presenting a simple example which illustrates the need of business driven web service selection mechanism when the requester has functional and non-functional requirements involving multiple QoS properties with varied preferences. Consider an online buying scenario in a shopping domain (typically e-commerce scenario). The buyers will have several requirements on the service quality, i.e. business qualities of the service. Some buyers prefer speedy delivery of the purchased item and some buyers may prefer free delivery, i.e. without service price. Also there are buyers who prefer the speedy delivery of the purchased item with a very low delivery charge. Such buyers will have preferences for the requested QoS properties. For example, the buyer might give higher preference to delivery time compared with the delivery price if the item to be purchased is an urgent requirement for him. Thus requesters will have different requirements on the service quality depending on the context or requester's behaviour. As a motivating example, consider the book buying scenario with the buyer's requirements on QoS as follows:

(a) A reputed book seller (ranked above 6 out of 10) who delivers book within 8 days with a delivery price less than \$10.
(b) Book seller who freely delivers a book within 15 days.

The buyer expects one of the requirements to be satisfied by the book seller and he gives a higher preference to QoS requirement (a). Assume that a web service discovery mechanism for the book purchase request finds multiple web services for the buyer. Now the problem is the selection of the best book seller web service for the buyer. To solve such problems, there is a need for the mechanism to select the best book seller service that meets the buyer's quality requirements and preferences. If the web service requester has several requirements involving different qualities with varying preference to each quality/requirement, then a need arises to identify:

- A common QoS vocabulary for web service requesters and providers for QoS-aware web service selection and publishing.
- A model to represent requester's complex requirements on service quality to choose desired web service.

- An extended web service architecture for QoS-aware web service selection and publishing.
- A selection and ranking mechanism for functionally equivalent web services based on requester's requirements on QoS.
- A mechanism to distinguish qualitatively competitive web services.
- A model to represent requester's alternative requirements on QoS and the selection mechanism.

In this article, the authors give the solutions to these key issues and the major contributions of this paper are:

- A QoS model (vocabulary) for business driven web services (i.e. web services) describing various QoS properties and their classification.
- A QoS constraint format to specify requester's complex requirements on QoS.
- A tree model and its XML equivalent to represent requester's complex QoS requirements involving multiple QoS properties having varied preferences.
- A QoS broker based architecture for QoS-aware web service selection and publishing.
- A selection and ranking algorithm to select best (qualitatively optimal) web service satisfying requester's QoS requirements and preferences.
- A proposal to rank the qualitatively competitive web services using web service provider qualities.
- The modelling of requester's alternative requirements on QoS.
- A web service selection mechanism for requester's alternative QoS requirements.

3. Quality of service (QoS) model for web services

Quality of service (QoS) in web services is a combination of several qualities (properties) of web service and it is a measure of how well a web service serves the requester. QoS is a measurable non-functional aspect of web service which can be used to discriminate functionally similar web services. In this section, the authors propose a generic (can be extended to include domain/service specific qualities) QoS model for web service selection, which groups the QoS properties based on the requester's selection point of view as business specific QoS properties, performance specific QoS properties and requester's response specific QoS properties.

3.1. Business specific QoS

The primary intention of a web service is to fulfil the requester's business requirement along with his quality requirements. Business specific QoS refers to a business value. The business specific QoS is crucial for both web service requesters and the providers since the requester and provider try to maximise their profit from the business. The authors identify four business specific QoS properties for web services.

3.1.1. Service charge (S_C)

The service charge is the amount of money the service requester has to pay to the service provider for the service consumption. A numeric metric in the range

[0.0, . . . , X | X is a finite real number] indicating currency can be used for the service charge.

3.1.2. Service withdrawal (S_W)

Service withdrawal is the time period, which commences after the receipt of web service request, during which the requester is allowed to cancel the service request without paying any fee. A numeric metric in the range [0.0, . . . , X | X is a finite real number] indicating hours/minutes can be used to represent service withdrawal value.

3.1.3. Service penalty (S_E)

The service penalty is the amount of money the service requester has to pay to the provider for the service cancellation after the service withdrawal. A numeric metric in the range [0.0, . . . , X | X is finite real number] indicating currency can be used to represent service penalty.

3.1.4. Service compensation (S_O)

Service compensation indicates the amount of money that will be refunded when the service provider cannot honour the committed service. A numeric metric in the range [0.0, . . . , X | X is a finite real number] indicating currency can be used to represent service compensation.

3.2. Performance specific QoS

The performance specific QoS properties refer to the performance of the web service system and are an indicator of how fast and how efficiently the web service system serves the request. The performance QoS properties are normally dependent on the static and dynamic behaviour of the web service system. The paper measures the performance of a web service system in terms of service time, service load, service readiness and service safety.

3.2.1. Service time (S_T)

Service time is the amount of time spent by the web service system to process the service request and to generate a positive response. Service time of a web service normally depends on the system configuration and the number of concurrent requests. A numeric metric in the range [0.0 . . . X | X is a finite real number] indicating minutes/hours/days can be used to represent the average (expected) service time.

3.2.2. Service load (S_L)

Service load is the maximum number of service requests that a web service system can process in a unit time (hour/minute) yielding a positive response. A numeric metric in the range [1, . . . , X | X is a finite positive integer number] indicating number of service requests can be used to represent the service load.

3.2.3. Service readiness (S_R)

Service readiness is the probability that a web service interface is available and ready for the access. It is the ratio of time period in which web service is ready for access to the total observation time period. A regional metric denoted by a numerical region [0.0, 1.0] can be used to represent the service readiness value.

3.2.4. Service safety (S_S)

Service safety refers to the security aspects of web services. It is measured based on the nature of mechanisms used for authentication, authorisation, non-repudiation, message integrity, message confidentiality and resilience for denial of service attacks. A graded metric denoted by integers [1, 2, 3, 4, 5] can be used to denote the service safety value, i.e. safety level.

3.3. Requester's response specific QoS

The authors consider two requester's response specific qualities (response specific QoS) which are estimated based on the requester's authentic feedback. Response specific QoS is measured by obtaining the requester's feedback after web service consumption under the assumption that the requester is willing to give his/her candid opinion/information when asked by the authentic third party and the information furnished by the requester can be trusted.

3.3.1. Service popularity (S_P)

Service popularity refers to the reputation and it is a measure of the service's trustworthiness. The value of service popularity is defined as the average ranking given by the requesters to a web service. A graded metric denoted by the integer values in an interval [1, 10] can be used to represent the service popularity.

3.3.2. Service deliverability (S_D)

Service deliverability is the probability that a web service request is positively responded to the requester within the maximum expected time frame (service time) as indicated in the service advertisement. A regional metric denoted by a numerical region [0.0, 1.0] can be used to represent the service deliverability.

The important feature of all the QoS properties is that they are either beneficial or lossy in nature. For a beneficial QoS property, higher value indicates better quality. The lower values of lossy QoS property indicate higher quality. For example, service time is lossy in nature since the lower the value of the service time means the better the web service. Similarly, the QoS property service readiness is beneficial, as the higher value indicates a better web service quality.

4. Requester's QoS constraints and modelling

The web service requester normally expects some requirements on QoS to be satisfied by the web services. The authors call these requirements on QoS properties QoS constraints. Formally, the paper defines QoS constraint as a relational expression

defined on a set of QoS properties. For example, a book buyer may have constraints like 'delivery should be within 4 days', 'price should not exceed \$50' etc. The QoS constraints are normally different for individual requester, which is dependent on the context and the requester's behaviour on QoS. For example, a book buyer normally looks for an inexpensive book supplier service that supports quick delivery. If the delivery is crucial for the buyer, then he prefers quick delivery. If the cost is the criterion, then he may prefer low delivery price (service price). Sometimes buyers look for a faster and cheap web service with preferences to delivery time and service price. Thus a web service requester can impose different constraints on several QoS properties with varying preferences for the web service selection.

4.1. QoS constraint types

The authors classify requester's QoS constraints based on the constraint structure as 'simple' and 'composite' QoS constraints. A simple QoS constraint normally deals with one QoS property. For example, a buyer might say, 'I need a book supplier for a service price less than \$50'. This is a simple QoS constraint which can be written as 'price < 50'. A simple QoS constraint takes the following format: Q_i rp V_i where Q_i refers to QoS property, rp refers to comparison operator ($<$, $>$, $=$, $\geq$, $\leq$) or membership operator (in) and V_i refer to expected single value or range of values for Q_i.

The authors further classify a simple QoS constraint as 'point' and 'range' QoS constraint based on the possible values of Q_i. A simple QoS constraint with equality operator ($=$) is called a point QoS constraint, i.e. Q_i of the point QoS constraint takes a single value. For example, the buyer might say, 'I need a book seller who delivers book in a day'. This is a point constraint and can be written as 'delivery time $= 1$'. A simple QoS constraint with range of values for Q_i is called range QoS constraint. The paper classifies range constraints based on the type of rp as 'implicit' and 'explicit' range QoS constraints. A simple QoS constraint with comparison operator from a set $\{<, >, \geq, \leq\}$ is referred to as implicit range QoS constraint. In implicit range QoS constraint, either minimum value of Q_i or maximum value of Q_i is implicitly related depending on the nature (beneficial/lossy) of Q_i. For example a buyer might say 'I need book seller with a reputation of more than 7 (out of ten)'. This is implicit range QoS constraint which can be written as 'reputation > 7'. Here the maximum value for reputation is 10 as defined in the QoS model. Thus the reputation value range is 8–10 where maximum value 10 is implicit value for Q_i. A simple QoS constraint with explicit minimum and maximum values for Q_i is referred to as explicit range QoS constraint. The explicit range QoS constraint is expressed using a membership operator in, minimum value of Q_i and maximum value of Q_i. For example, a buyer might say 'I need a book seller whose delivery price is between 4 and 8 dollars'. This is explicit range QoS constraint which can be written as 'price in [4–8]'.

A composite QoS constraint is composed of multiple simple QoS constraints using constraint composition operators AND and/or OR. For example, a buyer might prefer the book supplier service having service price less than \$3 and delivery time less than 3 days. This is a composite QoS constraint which can be represented as 'price < 3 AND delivery time < 3'. A composite QoS constraint takes the form C_1 cp C_2 cp C_3 cp, ..., cp C_P where, C_i refers to simple QoS constraint and cp denotes constraint composition operator AND/OR. In general a QoS constraint takes the form, (Q_i rp V_i) (cp (Q_j rp V_j))*. The web service requester can enforce either simple

QoS constraint or a composite QoS constraint during web service selection to choose desired web service.

4.2. QoS Constraint modelling

Consider the requester's QoS constraint which is defined on K (K > 0), QoS properties with preference to each simple and composite QoS constraint. The authors propose a tree structure and an XML equivalent to represent requester's QoS constraints with varied preferences.

4.2.1. AND-OR tree

An AND-OR tree (D'Mello and Ananthanarayana 2008b) is a non empty rooted tree of order (degree) N, with finite number of nodes and each node can contain zero or two or C ($2 < C \leq N$) child nodes. A node with no child is called a 'leaf' and the node with C child nodes is referred as 'internal' node. The internal node contains one item of information, i.e. AND or OR and the leaf node takes finite items of information.

The important property of AND-OR tree is that all the leaf nodes will be at the same level i.e. at level-0. Let H be the height of an AND-OR tree, then the level of root node will be H and the levels of internal nodes (except root) will be between 1 and H − 1. The level of any internal node (say X) can be computed as the maximum among the levels of child nodes +1. Figure 1 illustrates an important property of an AND-OR tree. The internal nodes are represented by ovals and the rectangles represent leaves. The nodes labelled with I, H, G and J are the internal nodes and the nodes A–F are leaves with finite information items. The node I represents the root which is AND node and the node H is of type OR. The height of a tree is 3 and the level of root node I can be calculated as, the maximum of level of node H and node A plus one i.e. max (2, 0) + 1 = 3.

4.2.2. Weighted AND-OR tree

A weighted AND-OR tree (D'Mello and Ananthanarayana 2008b) is an AND-OR tree in which every edge is labelled with a non-negative numeric value (weight/preference) such that, for any parent node, the sum of the edge labels of all its child nodes is equal to one, i.e. for any parent node P, with K ($2 \leq K \leq N$) child nodes ($C_1, C_2, \ldots, C_K$), the sum of edge weights W_{PCi} is equal to 1.

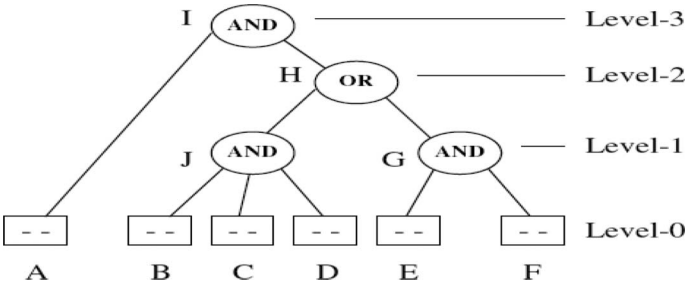

Figure 1. A simple AND-OR tree of height 3.

4.2.3. Quality constraint tree (QCT)

A quality constraint tree is a weighted AND-OR tree whose leaf nodes contains three or four information items i.e. leaf node contains QoS property (Q_i), comparison or membership operator (rp) and the expected QoS property value (V_i), or minimum and maximum values (V_{in}, V_{im}) for Q_i. The internal node refers to constraint composition operator (cp). The label (W_{XY}) on the edge between any two nodes X and Y represent 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 at the sub-tree rooted at node Y. The leaf node represents simple QoS constraint and any sub-tree rooted at the internal node represents composite QoS constraint.

As an illustration, consider the buyer's constraints on QoS for the online book buying as follows:

(1) Free delivery should be within 15 days without penalty for cancellations, having preferences 0.4, 0.4 and 0.2 for delivery time, delivery cost and penalty respectively.
(2) Delivery should be within 5 days, delivery cost should be less than \$10 and a reputed seller whose rating should be more than 5 out of 10, with preference 0.3, 0.3 and 0.4 for delivery time, delivery cost and reputation respectively.
(3) Highly reputed service with rating more than 7 (out of 10) and delivery charge should be less than \$6, with preference 0.6 and 0.4 for reputation and delivery charge.

The buyer expects best seller service which satisfies one of the QoS constraints out of three with equal preference to each constraint. The authors use the QoS vocabulary to express the constraints in a formal way and the representation is:

(1) Service time (S_T) $\leq$ 15 AND Service cost (S_C) = 0 AND Service penalty (S_E) = 0.
(2) Service time (S_T) $\leq$ 5 AND Service cost (S_C) $<$ 10 AND Service popularity (SP) $>$ 5.
(3) Service popularity (S_P) $>$ 7 AND Service cost (S_C) $<$ 15.

A quality constraint tree (QCT) for the requester's QoS requirements is depicted in Figure 2.

4.2.4. XML representation for QCT

Here the paper presents an XML representation for the requester's QoS constraints. If the requester is an agent program which interacts with the system hosting the selection mechanism then the requester's QoS constraints has to be supplied for the selection by encapsulating the XML equivalent of QoS constraint within the header of SOAP message. In an XML representation, the QCT is represented using a tag <QCT> with sub-tags <INT> and <LEAF>. The internal node is represented with a tag <INT> and it takes three attributes, namely type, weight and level. For any internal node X, the type attribute refers to the type of internal node (AND/OR), the weight attribute refers to label on the edge between node X and node P,

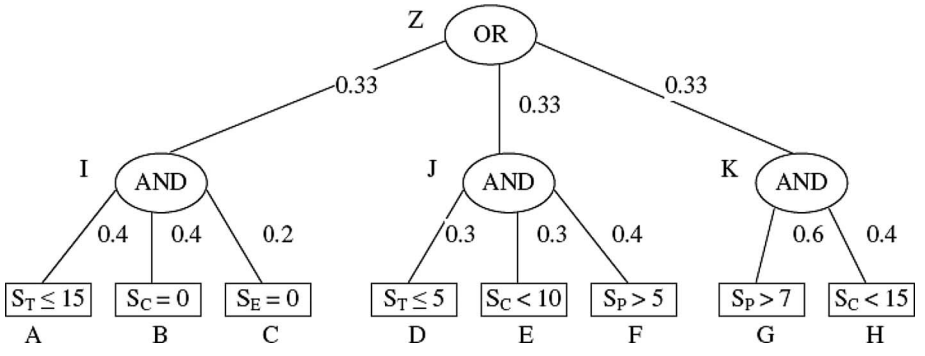

Figure 2. Quality constraint tree (QCT) for buyer's QoS requirements.

where P is proper (strict) ancestor of X and the level attribute refers to the level of the internal node. The leaf node is represented using tag <LEAF> that takes four attributes, namely quality, operator, weight and level. For any leaf node Y, the quality attribute refers to QoS property Q_i, operator attribute refers to the comparison operator rp; the weight refers to label on the edge between node Y and P where the node P is proper ancestor of node Y and the level attribute refers to the level of the leaf node. The expected value for Q_i, i.e. V_i, is placed in between <LEAF> and </LEAF> tags. The minimum and maximum value for Q_i (in case of range QoS constraints) is separated by the delimiter '-' (hyphen). The root node refers to whole QoS constraint, thus its weight attribute of is set to 1.

The QCT can be easily transformed into an XML equivalent by applying the procedures of basic XM processing. Figure 3 depicts the SOAP message containing an XML equivalent of QCT (Figure 2) for the QoS-aware web service selection.

5. A QoS broker based architecture for web services

The paper proposes the QoS broker based architecture for web services with an objective of selecting the best (qualitatively optimal) web service that satisfies the requester's QoS constraints and preferences. Figure 4 depicts the different roles and operations supported by the QoS broker (broker) based architecture. The architecture is based on the QoS vocabulary described as in Section 3 and assumes that both web service requesters and providers will use the same QoS vocabulary for QoS-aware web service selection and publishing.

5.1. Roles and operations

The authors define two additional roles to the conceptual web service architecture (Kreger 2001) called the broker and QoS registry. The authors also define two new operations, namely select and register. The broker is defined between registry services and the requester/provider. The broker facilitates the requester to specify his/her QoS constraints with preferences for web service selection and also assists the service providers to register their web services into registries. The broker also acts as a QoS certifier (Ran 2003) to issue the credentials for performance specific QoS properties registered by the providers. The QoS registry is a repository with business sensitive, performance sensitive and response sensitive QoS lookup and access support. The

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 xmlns:qos="http://www.nitk.ac.in/itdept/research/qualityinfo">
  <soap:header>
      <qos:QCT >
        <qos:INT type = "OR" weight = "1" level = '2'>
          <qos:INT type = "AND" weight = "0.33" level = "1">
            <qos:LEAF quality = "ST" operator = "≤" weight = "0.4" level = "0"> 15 </qos:LEAF>
            <qos:LEAF quality = "SC" operator = "=" weight = "0.4" level = "0"> 0 </qos:LEAF>
            <qos:LEAF quality = "SE" operator = "=" weight = "0.2" level = "0"> 0 </qos:LEAF>
          </qos:INT>
          <qos:INT type = "AND" weight = "0.33" level = "1">
            <qos:LEAF quality = "ST" operator = "≤" weight = "0.3" level = "0"> 5 </qos:LEAF>
            <qos:LEAF quality = "SC" operator = "<" weight = "0.3" level = "0"> 10 </qos:LEAF>
            <qos:LEAF quality = "SP" operator = ">" weight = "0.4" level = "0"> 5 </qos:LEAF>
          </qos:INT>
          <qos:INT type = "AND" weight = "0.33" level = "1">
            <qos:LEAF quality = "SP" operator = ">" weight = "0.6" level = "0"> 7 </qos:LEAF>
            <qos:LEAF quality = "SC" operator = "<" weight = "0.4" level = "0"> 15 </qos:LEAF>
          </qos:INT>
        </qos:INT>
      </qos:QCT>
  </soap:header>
  <soap:body>
     <find_service businessKey="*" generic="1.0" xmlns="urn:uddi-org:api" maxRows="100">
        <findQualifiers></findQualifiers>
        <name> bookseller </name>
     </find_service>
  </soap:body>
</soap:envelope>
```

Figure 3. An XML representation of QCT embedded within the SOAP header.

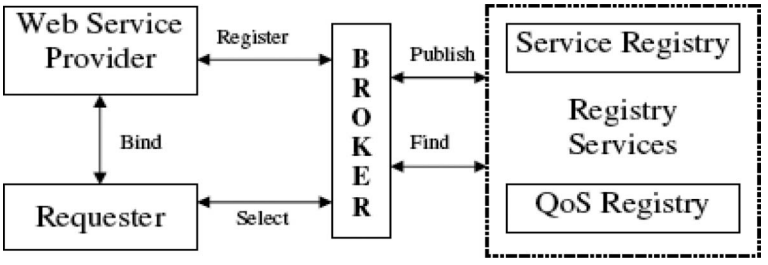

Figure 4. A QoS broker based web service architecture.

select operation is defined between the broker and requester, which selects the best web service based on the requester's QoS constraints. The register operation is defined between provider and the broker for QoS-aware web service publishing. The classical operations publish and find have specialised sub-operations, namely publish service, publish QoS, find service and find QoS. The publish service and publish QoS operation publishes service (including business information) specific and QoS specific information to the respective registries. The find service operation discovers the web services from the service registry through functionality matching (Kreger 2001, Riegen 2002). The find QoS operation retrieves QoS property values of web services to the broker/requester. From an architectural perspective, the broker is a middleware which can be implemented as a web service.

5.2. Component interactions

The authors design the broker with three components, namely service selector, service publisher and quality manager. For each component of the QoS broker, the

authors define a set of functions to fulfil the objective of QoS-aware web service selection and publishing.

5.2.1. *Service selector*

The main functionality of this component is to select the best web service satisfying the requester's QoS constraints and preferences. The responsibilities of this component include:

(1) Receiving information consisting of service functionality and QoS constraints from the requester.
(2) Construction of quality constraint tree for the requester's QoS constraints.
(3) Obtaining candidate web services (functionally similar) from the service registry.
(4) Retrieving QoS property values of candidate web services.
(5) Selection of the best web service from the candidates based on requester's QoS constraints and preferences.

We present the selection algorithm with an illustration in the next section.

5.2.2. *Service publisher*

The service publisher receives business, service and QoS specific information from the web service providers and publishes them into the corresponding registries. The service publisher publishes business/service specific information into service registry. The publisher component reads business specific QoS and performance specific QoS information from the providers, certifies the performance QoS and then publishes them into QoS registry. This component also facilitates the updating and deletion of business specific, service specific and QoS related information from the respective repositories. Figure 5 depicts the sequence of activities of QoS-aware web service publishing.

5.2.3. *Quality manager*

The main objective of this component is to estimate the requester's response specific QoS property values like service popularity and service deliverability through requester's feedback. The feedback from the requester is obtained under the assumption that the requesters are willing to return the QoS property values to the quality manager component, when asked by the broker, after the web service consumption and the received QoS values from the requester can be trusted. The broker architecture uses a feedback verification protocol (refer to Section 5.4) to obtain authentic feedback from the requesters.

5.3. *Web service publishing and selection*

Let $WS = \{funct, QS\}$ be the web service to be registered to the broker where *funct* represents functional information (including binding information) and QS signifies business and performance specific QoS information. The provider first registers web service information (functional and QoS) with the broker and then the broker

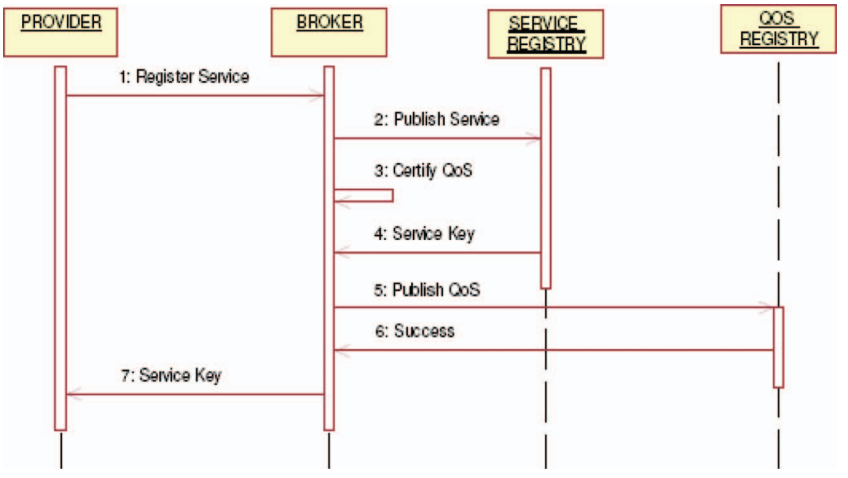

Figure 5. Sequence diagram depicting various activities of web service publishing.

publishes functional information into the service registry (UDDI registry). After successful publishing, the broker publishes business specific QoS into QoS registry and then certifies performance specific QoS. The QoS certification is done by the broker or it can be given to third party (the QoS certifier role proposed by Ran (2003)) for certification. Upon successful certification, the broker publishes performance specific QoS into QoS registry and sends the registration response to the provider.

Let $R = \{fn, QC\}$ be the request for QoS-aware web service selection from the requester to the broker where *fn* refers to requested functionality and QC represents the requester's QoS constraints (QCT). The broker finds functionally similar web services (candidates) through functionality matching techniques (Kreger 2001, Riegen 2002, Wang and Stroulia 2003) and then it finds the requested QoS properties of all candidates from the QoS registry. Now the broker executes the selection algorithm and returns the optimum service that satisfies QC to the requester. Figure 6 shows the sequence of activities involved in service selection mechanism.

5.4. *Feedback collection protocol*

The quality manager component of the broker receives feedback from the requesters to estimate the response specific QoS like service popularity. The broker collects the feedback from authentic requester who has consumed the service. In this sub-section, the paper presents a simple protocol which guarantees the authenticity of requester's feedback and prevents false and duplicate (clone) feedbacks from the requester. To achieve these objectives, the broker maintains a table called the 'authentication table' with the following fields: Req_Id, Prov_Id, WS_Id, Invoke_Id and Bind_Id where Req-Id refers to requester's identity (for example, username and password), Prov_Id refers to web service provider's business key, WS_Id is the web service key, Invoke_ID is the universally unique identifier (UUID) and Bind_Id is the universally unique identifier (UUID). Similarly, the web service provider maintains a table called the 'binding table' which stores the following information: Req_Id, Bind_Id, Invoke_Id and WS_Id. The protocol requires the implementation of public key

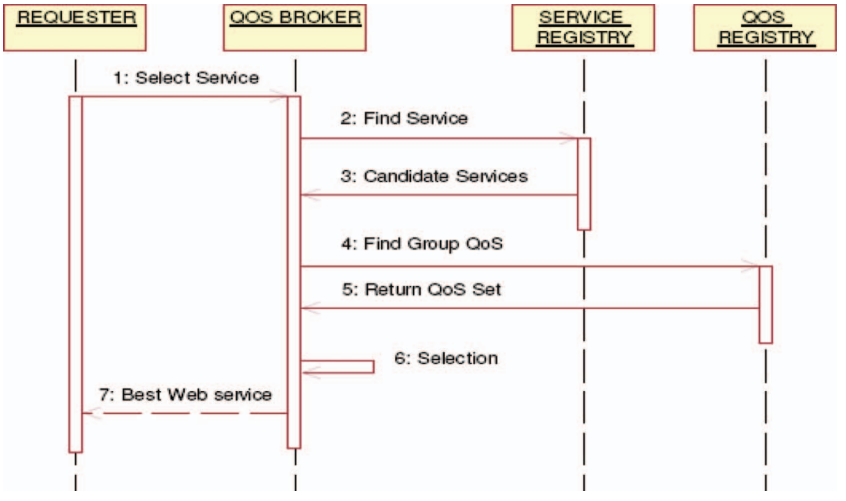

Figure 6. Sequence of activities of service selection.

infrastructure (PKI) at the broker, requester and provider's site for the secure interaction among these architectural roles. The protocol assumes the lossless and reliable communication link between the different architecture roles. The feedback received from the requester represents honest opinion about the QoS of consumed web service which can be trusted. Here the authors present the simple protocol which guarantees the authenticity of requester's feedback on service executions.

(1) The broker sends the best web service to the requester through a message Msg_BWS (Req-Id, WS_Id, Prov_Id, Bind_Info) where the Bind_Info refers to the web service binding information. Upon receiving this message, the requester obtains the binding information of the best web service.
(2) Requester generates a UUID called Invoke_Id for the web service to be consumed and sends the message Msg_Invoke (Invoke_Id, WS_Id, Req_Id) to the provider of the best web service.
(3) On receiving the message Msg_Invoke from the requester, the provider generates an UUID called Bind_Id and inserts the following information into the binding table: (Bind_id, Invoke_Id, Req-Id, WS_Id). Now the provider sends a message Msg_Ready (Bind_Id, Invoke_Id, WS_Id, Req_Id, Prov_Id) to the broker.
(4) On receiving the message Msg_Ready from the provider, the broker puts the entry into the authentication table with the following data (Req_Id, Prov_Id, WS-Id, Invoke_Id, Bind_Id). The broker now sends the message Msg_Bind (Bind_Id, Invoke_Id) to the requester. On receiving this message, the requester checks the Invoke_Id field of the message for authentication. On successful verification, the requester forwards the same message to the provider to initiate the service execution.
(5) On receiving the Msg_Bind from the requester, the provider executes the web service. On completion of service execution, the provider sends a response on service execution to the requester through the message Msg_Rsp (Bind_Id, Resp) where Resp is a vector consisting of service execution results/effects.

Now the provider deletes the binding table entry related to the service execution.

(6) After receiving response from the provider, the requester sends the feedback message (Msg_FB) about the service execution which is of the form: Msg-FB (Req_Id, Prov_Id, WS_Id, RFB) where RFB refers to feedback which includes the requester's experience (ratings, success/failure and others) about the service execution.

(7) After receiving the message Msg_FB, the broker updates the requester's response specific QoS of the best web service based on the ratings received for the various QoS parameters and removes the authentication table entry which is related to these interactions.

5.4.1. Fake feedback

A fake feedback refers to the feedback information obtained from the service requester about the specific web service without service consumption. The fake feedback is normally submitted by the requester either to boost or degrade the requester's response specific QoS of a specific web service. The proposed protocol prevents such feedback messages. On arrival of the requester's feedback (Msg_FB), the broker verifies the Invoke_Id and Bind_Id of the Msg_FB. If the Invoke_Id and Bind_Id are found in the authentication table then the quality manager of the broker uses the feedback information (RFB) to estimate the requester's response specific QoS properties otherwise the arrived Msg_FB is treated as a fake feedback message.

5.4.2. Detection of clone (duplicate) feedback

A clone feedback is the duplicate feedback information obtained from the web service requester about the specific web service after the service execution. Once the Msg_FB is verified for the authentication by the broker, the broker deletes the entry related to the completed interaction from the authentication table. If the broker receives a duplicate feedback then the Invoke_Id and Bind_Id present in the message are unavailable in the authentication table and the arrived message is treated as a fake feedback message. Thus the protocol is resilient to both fake and multiple feedbacks for a single instance of web service execution.

5.5. Resource and latency overhead

The web service publishing involves six message exchanges between various roles of the broker based architecture. Most of the activities of web service publishing are performed in sequence except the two activities certify QoS and service key message. These activities are performed in parallel which may reduce significant latency overhead involved in the interactions. Figure 7 shows the activity diagram involving parallel activities of the service publishing.

The web service selection requires six messages to be exchanged between various roles of the architecture. The activities involved in the selection are executed in sequence. In order to improve the business and to be in competition with other business providers, the provider of the web service has to advertise the QoS of web service which involves an additional four message exchanges and the latency as compared to the conceptual web service architecture. To obtain the right web

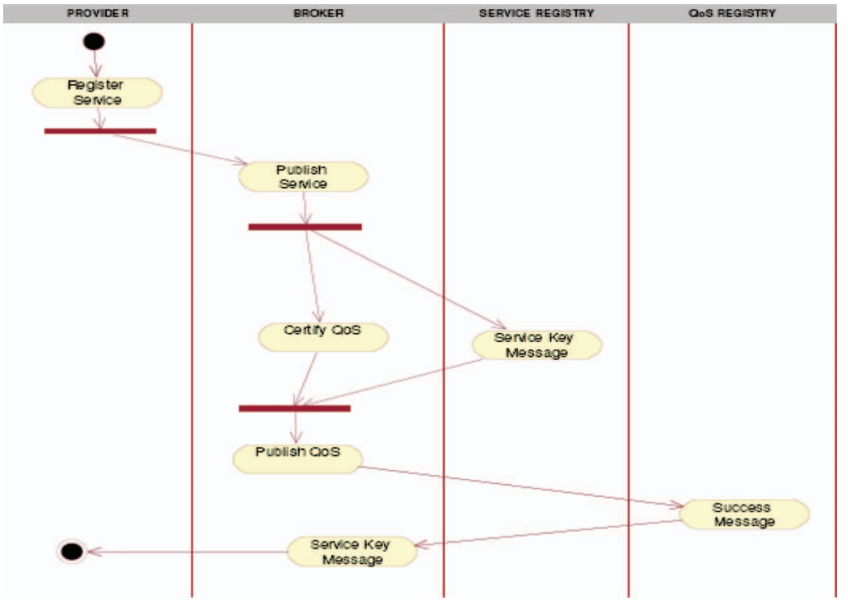

Figure 7. The swim lane activity diagram of service publishing.

service from functionally similar web services, the requester has to send a query through the broker which involves an additional four messages and the associated latency. Using the broker-based architecture for selection, the requester is freed from the burden of selecting the good service. In case of business-to-business (B2B) service interactions, the additional latency involved in the selection mechanism is negligible as compared to the long running process execution time. Similarly the feedback collection protocol requires six additional messages to be exchanged between different roles of the architecture to estimate the new values for the requester's response specific QoS properties of consumed web services. The activities of feedback collection protocol are carried out sequentially in order to prevent the fake or duplicate feedbacks.

6. The web service selection mechanism

The authors propose web service selection and ranking algorithm which algorithm takes a QCT T of height H and the candidate web services as an input and gives the best web service that satisfies requester's QoS constraints and preferences. The algorithm uses a table named VAL to store the real-valued numbers (processed QoS values) for candidate web services selected at any tree node. The row index of a table refers to a tree node and the column index indicates the web services. The algorithm traverses QCT in level order fashion (level 0 to level H) and treats leaf and internal nodes in a different manner. At leaf nodes, the algorithm performs the following three actions: (1) filtering (2) scaling and (3) ranking. In filtering phase, the web services that satisfy simple QoS constraints defined at the leaf nodes are selected. The scaling phase normalises the QoS values of selected web services to a non-negative real valued number in an interval [0, 1] using min-max normalisation technique (Taher *et al.* 2005), where the higher normalised values represent higher quality. In

ranking phase, the normalised values are multiplied with the weight (preference given to the QoS constraint) to get new values representing scores for the web services.

At the internal nodes, the algorithm performs two actions: (1) filtering and (2) ranking which are dependent on the type of node (AND/OR). In the filtering phase, if the node is of type 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 the ranking phase, if the node is AND, then the score of selected web service is computed as the sum of scores of selected web services at its child nodes multiplied with the weight of sub-tree rooted at AND node. If the node is of type OR, then the score of selected web service is multiplied with the weight of sub-tree, rooted at that node. After ranking the web services at the root node, web services are sorted in the descending order of their score and the first web service is returned to the requester as a best (qualitatively optimal) web service. The detailed algorithm (pseudo code) is presented in Figure 8.

The proposed service selection algorithm first filters the web services based on the QoS constraints defined at leaf nodes. The QoS values of selected web services are then normalised, where higher normalised values indicate the better quality. At internal nodes, the scores of selected web services are multiplied with requester's preference for a given simple/composite QoS constraint. At any node, the selected web service indicates its suitability for the QoS constraint defined at that node. After sorting web services at the root, the first web service becomes the optimum web service for the requester.

6.1. Algorithm analysis

In this section, the paper analyses the time and space complexity of the selection algorithm. Let M be the candidate web services, H be the height of QCT and N be the degree of QCT. The maximum number of nodes present in any QCT of height H is $(N^{H+1} - 1)/(N - 1)$. The ranking operation is the basic operation of the selection algorithm which is performed to get the score (rank) of web service at each node of QCT. At the worse case, all M candidate web services are selected at each node and the basic operation is executed for all M web services which results in a total of $M \times (N^{H+1} - 1)/(N - 1)$ number of basic operations. Thus worst case time complexity of the selection algorithm is $\theta(M \times (N^{H+1} - 1)/(N - 1))$. The best execution of the algorithm involves a QCT of height zero (single leaf) and one web service selected for the leaf which results in a single basic operation. Thus best case time complexity is $\theta(1)$.

The algorithm uses an additional space through the use of table called VAL of size $(N^{H+1} - 1)/(N - 1) \times M$. Let B be the maximum number of bits required to represent the score of a web service at a given node of QCT. The worst case space complexity of the selection algorithm is $\theta(M \times (N^{H+1} - 1)/(N - 1) \times B)$ and the best case space complexity is $\theta(B)$.

6.2. Illustration for the selection algorithm

Consider the book buyer's QoS constraints for the best book supplier service presented in Section 4.2.3 (refer to Figure 2). Assume that 10 book supplier services ($S_1, S_2, \ldots, S_{10}$) are found for the buyer from web service registry. The QoS properties for these candidate web services are as in Table 1. The algorithm first

Algorithm 1.

Input: QC tree T of height H, having S number of nodes
Candidate Web services List, WS[M] = {WS_1, WS_2, WS_3… WS_M}

Output: Best Web service which satisfies requester's QoS constraints and preferences

1. Initialize all the table entries VAL [i, j] to NULL, for i =1 to S and j =1 to M

2. For each QC tree node (X) of the form, Q cp V at level-0 (i.e., leaf node) do the following
   (i) Retrieve QoS property values for Q for all Web services WS[j], j ∈ {1 to M}, which satisfies the simple QoS constraint defined on QoS property Q at X
   (ii) Normalize the QoS property values based on nature of QoS property and store values in a table VAL as follows

$$VAL[X, j] = \frac{Qval - Qmin}{Qmax - Qmin} \quad \text{if Q is beneficial and Qmax - Qmin} \neq 0$$

$$VAL[X, j] = \frac{Qmax - Qval}{Qmax - Qmin} \quad \text{if Q is lossy and Qmax - Qmin} \neq 0$$

$$VAL[X, j] = 1 \quad \text{if Qmax - Qmin=0}$$

   Where,
   VAL[X, j] – Normalized QoS property (Q) value for Web service WS[j]
   Qval – QoS property (Q) value for Web service WS[j]
   Qmax – Maximum value for Q
   Qmin – Minimum value for Q

3. For level = 1 to H do
   For each node (X) do the following
   (i) If X is AND node then
   (a) Select the Web services those present in *all C* child nodes of X
   (b) For each selected Web service (WSj) do

$$VAL[X, j] = \sum_{i=1}^{C} Wxi * VAL[i, j]$$

   (ii) If X is OR node then
   (a) Let U = {U_1, U_2, U_3…U_C} be the C child nodes of X
   (b) For each child node U_i in U do
   For each Web service WS_j at Ui do
   If (VAL[X, j] < W_{XUi} * VAL [U_i, j]) then
   VAL [X, j] = W_{XUi} * VAL [U_i, j

4. Sort the Web services WS_j in the non increasing order of values VAL[S-1, j] where S-1 is the index of root node and j is the index for Web service WS_j

5. Return first Web service from the sorted list as a best Web service

Figure 8. QoS aware web service selection algorithm.

filters and ranks candidate web services based on the simple QoS constraints defined at the leaf nodes. For example, at node B, the web services S_4 and S_8 are selected for QoS constraint '$S_C = 0$'. Now, the QoS values of the selected web services are normalised using min-max normalisation technique and then multiplied with the QoS constraint preference. For example, the score for S_4 and S_8 at node B is 0.4. Table 2 shows the scores for selected web services at the leaf nodes of the QCT. The table entry '–' (dash) implies that the web service is not selected for the node.

Now, the algorithm takes internal nodes at level-1, i.e. AND nodes. The web service found in all child nodes of AND node is selected and the new score is computed by adding the score of web service at child nodes and multiplying the sum with the node preference. For example, the only web service S_4 is found common in the child nodes of node I. Thus S_4 is selected for node I with new computed value

Table 1. Candidate web services and their QoS property values.

Web service	S_T (days)	S_C ($)	S_E ($)	S_P (1–10)
S_1	4	5	3	8
S_2	8	7	2	5
S_3	3	4	5	8
S_4	12	0	0	4
S_5	4	3	4	5
S_6	15	9	1	9
S_7	18	10	7	3
S_8	7	0	2	6
S_9	1	12	0	7
S_{10}	2	18	3	8

Table 2. Scores of candidate web services at the leaf nodes.

Web service	Node A	Node B	Node C	Node D	Node E	Node F	Node G	Node H
S_1	0.314	–	–	0.2	0.133	0.266	0.0	0.233
S_2	0.200	–	–	–	0.066	–	–	0.166
S_3	0.342	–	–	0.1	0.166	0.266	0.0	0.266
S_4	0.085	0.4	0.2	–	0.300	–	–	0.400
S_5	0.314	–	–	0.0	0.200	–	–	0.300
S_6	0.000	–	–	–	0.00	0.400	0.6	0.100
S_7	–	–	–	–	–	–	–	0.066
S_8	0.228	0.4	–	–	0.300	0.000	–	0.400
S_9	0.400	–	0.2	0.3	–	0.133	–	0.000
S_{10}	0.371	–	–	0.0	–	0.266	0.0	–

0.226 ((0.085 + 0.4 + 0.2) × 0.33). For node J, the web services S_1 and S_3 are selected and the new scores for these web services are 0.198 and 0.176 respectively. Similarly for node K, the web services S_1, S_3 and S_6 are selected and their new score is 0.074, 0.088 and 0.363 respectively. At level 2 of the QCT (OR node), the web services S_1, S_3, S_4 and S_6 are selected from the nodes I, J, and K. The new scores for selected web services at OR node are 0.198, 0.176, 0.226 and 0.363 respectively. Now the web services are sorted in the descending order of their scores and the first web service, i.e. S_6, is selected for the requester as a best web service. Figure 9 shows the selected web services at the various tree nodes and the computed QoS score.

7. Qualitatively competitive web services and the selection mechanism

Let T be the QCT representing requester's QoS constraints. Let K be the number of web services selected and ranked for the requester based on QCT. After sorting the web services in the descending order of their scores, if the first X ($1 < X \leq K$) web services are found to have the same score, then such web services are called *qualitatively competitive web services* with respect to requester's QoS constraints. If such web services are found during web service selection, then which web service is to be selected for the requester? There is a need to identify the mechanism to

distinguish and rank the qualitatively competitive web services. Here the authors present an illustration for the problem by taking the QoS constraints as follows: $S_C < 50$ OR $S_P > 6$ with an equal preference for S_C and S_P. Assume that the service registry returns *three* web services through functionality matching. The S_C and S_P values of these web services are given in the format {web service key (Id), S_C, S_P} as follows: {S_1, 40, 9}, {S_2, 90, 4}, {S_3, 20, 7}. Figure 10 depicts the QCT representation for the QoS constraint illustrating the problem.

The web service selection mechanism described in the previous section (Section 6) selects *two* web services for the requester and computes same score, i.e. 0.5 for both S_1 and S_3. In such a scenario, a need arises to resolve the tie that may arise among qualitatively competitive web services.

7.1. Web service provider qualities

The authors define three qualities for web service providers. These qualities include provider popularity, provider existence and provider size. The main objective of defining web service provider qualities is to distinguish functionally similar and qualitatively competitive web services during web service selection.

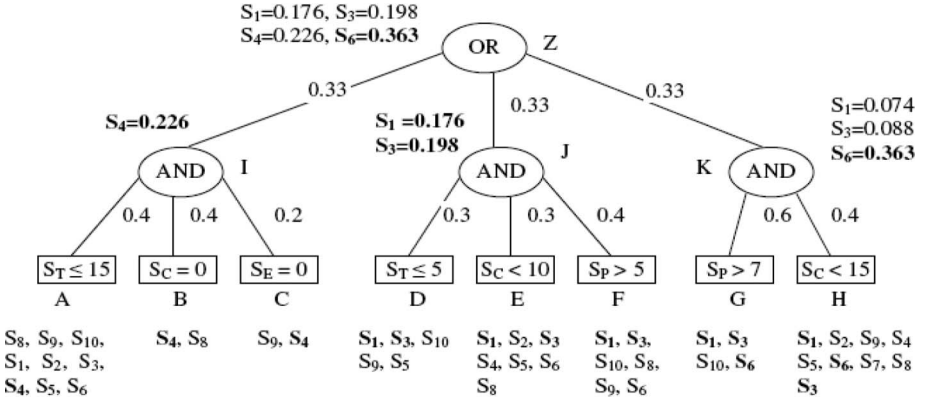

Figure 9. The trace of web service selection algorithm.

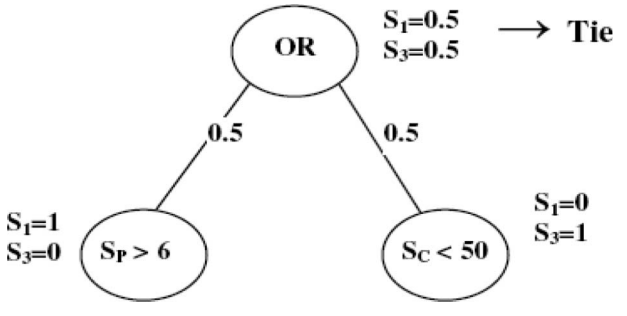

Figure 10. Qualitatively competitive web services.

7.1.1. Provider popularity (P_{SP})

Let Z be the number of web services hosted by the web service provider 'SP'. Let S_{Pi} be the service popularity of *i*th web service S_i. The popularity of web service provider (P_{SP}) can be estimated as the average of service popularity (S_{Pi}) values of Z web services.

7.1.2. Provider existence (E_{SP})

The provider existence of web service provider can be measured in terms of months/years. Provider existence is measured based on the date of business entity publication in the service registry (UDDI).

7.1.3. Provider size (S_{SP})

The provider size of web service provider is defined as the total number of web services hosted and published by the web service provider. This is an indicator of the provider's capability of hosting varieties of web services.

If the web service selection mechanism finds qualitatively competitive web services for the requester, then these web services are rearranged (sorted) based on the values of web service provider qualities in the following order: $P_{SP} << E_{SP} << S_{SP}$. The qualitatively competitive web services are first ranked based on the values of provider popularity. If multiple web services are found with same provider popularity, then the provider existence is used for the distinction assuming that the older service provider is more reliable than the newer provider. Again, if multiple web services are found with equal provider existence, then the provider size can be used as a tie breaker. Finally, if the qualitatively competitive web services are found with same provider qualities, then one among them (random selection) is selected for the requester.

8. Requester's alternative QoS constraints and modelling

The common tendency of the requester is to enforce a strong constraint on QoS for web service selection. In such cases, the web service selection algorithm described in Section 6 may not find a suitable web service for the requester, i.e. there is no web service available that meets requester's strong QoS constraints. Now the requester has to submit a slightly weak QoS constraint to get a good web service. The requester has to repeat sending weak QoS constraints until he gets a web service of his choice or he is not willing to have the lower quality web service. This process is time consuming for the requester and also a burden for the broker as it has to execute the selection algorithm several times. To avoid these problems, the requester can submit a set of alternative QoS constraints in the order of diminishing preferences for the web service selection. In this section, the authors propose a tree structure to represent requester's alternative QoS constraints and the associated selection mechanism.

8.1. Alternative QoS constraint modelling

Consider $S = \{QC_1, QC_2, \ldots, QC_R\}$ to be the set of R QoS constraints of a requester. Let QC_1 be the strong QoS constraint and QC_2 to QC_R be the alternative

QoS constraints in the order of diminishing preferences. The authors extend the QoS constraint structure capability to specify requester's set of alternative QoS constraints. The alternative QoS constraints are represented as QC_1 XOR QC_2 XOR, ..., QC_R. The authors also extend the quality constraint tree (QCT) structure to represent requester's set of alternative QoS constraints having diminishing preferences. Let QCT_1, $QCT_2, \ldots, QCT_R$ be the quality constraint trees for R alternative QoS constraints. The authors introduce a new node called XOR and then attach R QCTs to the XOR node. Now the XOR node becomes the root for all quality constraint trees. A real number in an interval [0, 1] is assigned to each edge, present between XOR node and roots of QCTs, such that the sum of weights of edges between XOR node and the roots of QCTs is equal to 1. Let $E_1, E_2, \ldots, E_R$ be the edges between XOR node and roots of R QCTs. The weight (W_i) on the edge E_i is calculated as: $2*(R - i + 1)/R(R + 1)$.

As an example, consider the requester's strong and weak QoS requirements, i.e. one alternative QoS constraint, as follows:

(1) Web service with service time equal to 1 and service charge less than \$3.
(2) Web service with service time less than or equal to 3 and service charge less than \$5 and service popularity above 6 (out of 10).

The extended QCT for these alternative QoS constraints is depicted in Figure 11.

8.2. *The selection mechanism*

The proposed web service selection mechanism takes extended QCT, T and the candidate web services as an input and gives the best web service as output based on the requester's set of alternative QoS constraints. The service selector component of the broker reads the extended QCT and executes the algorithm by taking the root (XOR node) of T. Now, the algorithm finds the sub-tree having the highest preference and sends the sub-tree representing strong QoS constraint to the Algorithm 1 along with the candidate web service list. If the algorithm returns no service, then the algorithm takes the sub-tree having second highest preference and sends it to Algorithm 1. This process is repeated until a web service is found for the

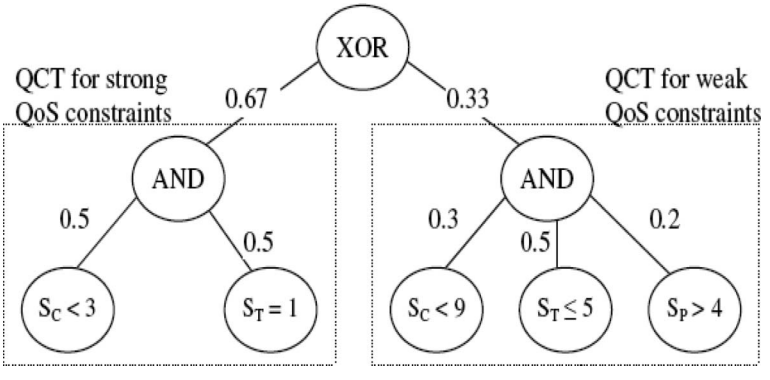

Figure 11. An extended QCT for alternative QoS constraints.

QoS constraint or all the alternatives are tried. The extended web service selection algorithm is presented in Figure 12.

9. Implementation and experiments

In this section, the authors compare and analyse the proposed web service selection mechanism with the closely related works (classical mechanisms) by conducting analytical experiments. The authors also present the QoS broker based web service architecture implementation details with experimentation results. The authors discuss the strength, weakness, scope and importance of the web service selection mechanism.

9.1. Analytical experimentation

Here the authors compare the results of *six* QoS aware web service selection mechanisms with the proposed selection mechanism by taking a suitable selection scenario. Consider the following six online seller services which differ in QoS discovered from the service registry through functionality matching. Table 3 shows the values of a few QoS properties of web services.

The authors analyse the results of the selection mechanisms (including the one proposed) for the following three different QoS constraints:

- Constraint-I: Service charge <25;
- Constraint-II: Service popularity ≥ 5 AND service time ≤ 4 with equal preference to QoS properties;

Algorithm 2.

Input: Extended QCT, Candidate Web service list (WS)
Output: Best Web service

Let 'E' be the root (XOR node) of extended QCT
Let $Y_1, Y_2 \ldots Y_R$ be the child nodes of 'E' in the decreasing order of edge weight (W_{SY})
For $Y = Y_1$ to Y_R do
{
  Let T_Y be the QCT rooted at Y
  BWS = Algorithm 1(T_Y, WS)
  If BWS $\neq$ Nil then
    Return BWS to the requester
  End If
}

Figure 12. An extended web service selection algorithm for the alternative QoS constraints.

Table 3. QoS property values of web services.

Web service	Service charge	Service penalty	Service readiness	Service popularity	Service time
WS_1	50	10	0.7	5	2
WS_2	30	5	0.5	4	7
WS_3	40	8	0.4	7	6
WS_4	20	15	0.8	3	5
WS_5	35	20	0.7	6	3
WS_6	10	25	0.3	8	4

- Constraint-III: (popularity $\geq$5 AND service time $\leq$4) OR service charge $<$25 with equal preference to QoS properties and sub-constraints.

Table 4 gives the scores and ranks assigned by the selection mechanisms to different web services. From the table, it is observed that for QoS constraint-I the web service WS_6 is selected as a best web service in most of the approaches except for Euclidian distance based method (Taher *et al.* 2006). This is because Euclidian distance based method will not consider the QoS property type (beneficial or lossy) while computing the rank. The mechanism proposed in Sodki *et al.* (2008) computes the QoS counts from the selection matrix to rank the web services which results in a tie between two web services, i.e. WS_4 and WS_6.

The selection mechanism discussed in Kerrigan (2006) does not handle QoS constraints defined on the multiple QoS properties (QoS constraint-II). The proposed selection mechanism and the Euclidian distance based method (Taher *et al.* 2005) selects WS_5 as a best web service for QoS constraint-II. This is due to the fact that the service popularity and the service time values of WS_5 are equidistant from the requested values. The selection mechanism discussed in Liu *et al.* (2004) selects WS_1 as a best web service. This mechanism employs complex QoS matrix operations and normalisations. In literature, the authors did not find a mechanism to handle

Table 4. QoS scores and ranks assigned by the different selection mechanisms.

QoS constraint / Selection methods	QoS constraint-I (Selected web service, score, rank)	QoS constraint-II (Selected web service, score, rank)	QoS constraint-III (Selected web service, score, rank)
Filtering and ordering (Kerrigan 2006)	(**WS_6**, 10, 1) (WS_4, 20, 2)	Not applicable	Not applicable
QoS group based normalisation (Liu *et al.* 2004)	(**WS_6**, 1, 1) (WS_4, 0.857, 2)	(**WS_1**, 0.37, 1) (WS_5, 0.33, 2) (WS_6, 0.3, 3)	Not applicable
Euclidian distance (Taher *et al.* 2005)	(**WS_4**, 0.33, 1) (WS_6, 1, 2)	(**WS_5**, 0.6, 1) (WS_1, 1, 2) (WS_6, 1, 2)	Not applicable
QoS discrepancy (Lo and Wang 2007)	(**WS_6**, 1.98, 1) (WS_4, 0.66, 2)	(**WS_6**, 0.099, 1) (WS_1, 0.083, 2) (WS_5, 0.074, 3)	Not applicable
Selection matrix (Sodki *et al.* 2008)	(**WS_4**, 1, 1), (**WS_6**, 1, 1) (WS_1, 0, 2), (WS_2, 0, 2) (WS_3, 0, 2), (WS_5, 0, 2)	(**WS_1**, 2, 1), (**WS_5**, 2, 1) (**WS_6**, 2, 1), (WS_1, 0, 2) (WS_2, 0, 2), (WS_3, 0, 2)	Not applicable
Simple additive weight (Jeager and Goldmann 2006)	(**WS_6**, 1, 1) (WS_4, 0, 2)	(**WS_1**, 0.25, 1) (**WS_6**, 0.25, 1) (WS_5, 0.21, 2)	Not applicable
Constraint tree (proposed method)	(**WS_6**, 1, 1) (WS_4, 0, 2)	(**WS_5**, 0.42, 1) (WS_1, 0.25, 2) (WS_6, 0.25, 2)	(**WS_6**, 0.5, 1) (WS_5, 0.21, 2) (WS_1, 0.125, 3) (WS_4, 0, 4)

requester's QoS constraints defined on multiple QoS properties involving AND and OR operators. The proposed selection mechanism identifies WS_6 as a best web service for QoS constraint-III. If the heuristic rules (proposed method to handle QoS constraints with OR operators) are applied to the classical mechanisms, then the web service WS_6 is selected. The use of such rules may not guarantee the delivery of correct result in all circumstances since the classical mechanisms handle the individual QoS constraints independently. Moreover the selection mechanism has to be executed multiple times depending on the QoS constraint structure. The proposed selection mechanism yields a best web service as it always tries to meet the requester's demands put on the service quality (QoS), i.e. at any QCT node the score of the web service score refers to the level of satisfaction of the requester's QoS requirement.

9.2. Implementation

The QoS broker based architecture is implemented on the Windows XP platform using Microsoft Visual Studio .NET development environment and Microsoft visual C# as a programming language. The authors use SAP UDDI V3 Test Public Business Registry as web service registry for the implementation. To enable the interaction between .NET program and the UDDI-compliant server, the authors use Microsoft UDDI .NET 2.0 Beta 1 SDK. To retrieve the functionally similar web services (candidate web services) from the service registry, the authors use SAP UDDI V3 Test Public Business Registry enquiry API (http://uddi.sap.com/uddi/api/inquiry). The QoS registry is implemented as web service which is accessible for the QoS broker and a separate graphical interface is created for the requester to retrieve QoS values of web services.

9.2.1. QoS broker implementation

The QoS broker functionality is implemented in three different modules. They are: service selection module, service registration module and response collection module. The interaction with these modules is accomplished by providing a separate interface form which is triggered through the menu interface.

The service selection module reads the requested functionality from the web service requester through an interface form. The module also retrieves and displays the functionally similar/same web services from the SAP business registry through the service discovery. If the discovery process results in a single web service, then the discovered web service is selected as a best web service. The discovery of multiple web services results in a new interface form which facilitates the requester to impose QoS constraints in the form of QCT or extended QCT for the selection. The requester's QoS constraints (QCT) are supplied to the selection mechanism as follows. In the interface form, the size of the alternative QoS constraints i.e. R (*two* alternatives are allowed for a strong constraint, i.e. $R = 3$) is supplied. The separate interface forms are created on the fly to read R QoS constraints (alternative QoS constraints) from the requester. The authors restrict the nesting depth of each QoS constraint to four (QCT degree 4). The simple QoS constraints are read in the following form: {operator, QoS property, value, weight} and the composite QoS constraints are read in the form: {operator, C_1, C_2, C_3, C_4, weight} where C_1, C_2, C_3, C_4 are the simple QoS constraint references and the weight refers to requester's preference. Now the broker retrieves the required QoS property values from the QoS

registry and executes the selection algorithm. After completion of the selection mechanism, the service selection module displays the web services in the descending order of the satisfaction of the requester's QoS constraints and preferences.

The service registration module provides three different user interface forms for the service registration, i.e. to publish business, service and QoS information. The service provider is allowed to publish his business information to the SAP business registry through the interface form. The successful registration of business information into SAP business registry is responded through the message which consists of business key of the provider. The QoS aware provider of web service can publish the service specific information into the business registry through an interface form. The business specific QoS and performance specific QoS is published into the QoS registry through an interface form. The successful advertisement of QoS-aware web service is acknowledged by the broker to provider, through the message consisting of service key of the published web service. The service registration module also provides interface forms to edit the business specific, service specific and QoS specific information of web services by the respective providers.

The response collection module consists of feedback form which facilitates the requester to give his honest opinion, i.e. information about the web service after the service consumption. The feedback information includes: success/failure information, the requester's observed values for business and performance QoS properties, etc. After the submission of feedback information, the response collection module updates the requester's response specific QoS property values like service popularity and service deliverability of the consumed web service.

9.2.2. QoS registry implementation

The QoS registry facilitates both business specific and performance specific QoS search options for a single web service and for a set of web services. The QoS registry is implemented as a web service which in turn interacts with the Microsoft SQL Server 2000 database which is responsible for the storage of QoS property values of various web services. The QoS registry provides both programmatic access and interface form (manual) based access to the QoS values of various web services.

Interface forms are designed to perform business specific, performance specific and requester's response specific QoS search for a given web service. Similarly, the retrieval of multiple QoS property values of the different web services through a single query (Find Group QoS) is allowed. A single QoS data table is maintained at the SQL Server database which is ordered based on the web service key. The structure (QoS property fields and their bounds) of the QoS data table is presented in Table 5.

9.2.3. Experiments

The authors have conducted several experiments to verify the implemented selection mechanism. A simple experiment (without alternative QoS constraints) is described below.

Required service: Book seller service
QoS constraint: $S_C < 800$ AND $S_P > 4$ with equal preference to S_C and S_P

Tables 6 and 7 show QoS constraint inputs i.e. simple and composite QoS constraints. Table 8 shows the candidate web services retrieved from SAP business test registry and their published QoS property values.

The web service selection algorithm selects the web service WS_2 (BookService) for the requester as a best (qualitatively optimal) web service.

Another experiment is conducted to select the best air travel reservation service for the passenger/traveller with the following alternative QoS constraints in the order of diminishing preferences:

(1) Service charge (S_C) < 3 AND service popularity (S_P) > 8 with equal preference to service charge and service popularity.
(2) Service penalty (S_E) < 5 AND Service charge (S_C) < 6 with equal preference to service penalty and service charge.

The passenger gives preference to QoS constraint (1) over QoS constraint (2). The SAP business test registry returns two web services (candidates) for the air travel

Table 5. QoS properties and their range of values.

QoS property	Range of values	Data type
Service charge	0–1000	Double
Service penalty	0–500	Double
Service compensation	0–1500	Double
Service withdrawal	0–5	Integer
Service time	0–10	Integer
Service load	0–1	Float
Service readiness	0–1	Float
Service safety	1–5	Integer
Service popularity	1–10	Integer
Service deliverability	0–1	Float

Table 6. Simple QoS constraints.

No.	Operator	QoS property	Value	Weight
1	<	S_C	800	0.5
2	>	S_P	4	0.5

Table 7. A composite QoS constraint.

No.	Operator	Constraint reference	Constraint reference	Weight
3	AND	(1)	(2)	1

Table 8. Candidate web services and their QoS values.

Web service	Service charge (S_C)	Service popularity (S_P)
BookShopping (WS_1)	785	5
BookService (WS_2)	587	9

reservation web service discovery request. The service charge (S_C), service penalty (S_E) and service popularity (S_P) values for these candidate web services are given in Table 9.

The selection algorithm finds no suitable web service for the QoS constraint (1) but finds AirTravelReservation (WS_1) as a suitable web service for the substitute QoS constraint, i.e. QoS constraint (2). Thus the web service AirTravelReservation (WS_1) becomes the best air travel reservation web service for the traveller.

9.3. *Analysis of the approach*

The proposed web service selection mechanism selects the best web service based on the requester's complex QoS constraints defined on the multiple QoS properties involving AND and OR operators. In literature, the selection mechanisms are defined based on the QoS constraints involving only AND operators (Liu *et al*. 2004, Taher *et al*. 2005). If the requester imposes a QoS constraint involving OR operators with AND operators then the classical (existing) selection mechanism has to be executed many number of times and the heuristic rule has to be applied to find the best web service. Moreover the number of executions is dependent on the number of sub-constraints involved in the QoS constraints. Thus the repetitive execution of the selection algorithm is always dependent on the QoS constraint structure. For example, in Figure 2 the QoS constraint consists of a single OR operator involving three sub-constraints. The proposed web service selection algorithm is suitable for the QoS constraints involving AND/OR operators, since the algorithm makes a single scan of the QoS constraint. Let $C_1, C_2, \ldots r, C_N$ be the simple QoS constraints (N is the nesting depth or the degree of QCT). The number of executions of classical and proposed selection algorithms for the various QoS constraint structures is presented in Table 10.

From the above analytical experiments on the number of selection algorithm executions, it is observed that for a given QoS constraint structure involving alternate AND and OR operators in the QoS constraint hierarchy, the proposed selection algorithm is executed only once whereas the execution of classical selection algorithm is exponential. For example, the classical selection algorithm has to be executed 5^4 times for the QoS constraint (QCT) of height 4 having degree 5 (alternate AND and OR nodes at tree levels). Figure 13 shows the graph representing the selection algorithm execution counts for the various QCT (degree 3) structures.

9.3.1. *Expressivity of the QoS constraint structure*

The QoS constraint structure defined in Section 4 provides the structured format to represent the requester's variety of requirements on the web service QoS. The proposed structure is capable of expressing requester's requirement on a single QoS

Table 9. Candidate web services and their QoS values.

Web service	Service charge (S_C)	Service penalty (S_E)	Service popularity (S_P)
AirTravelReservation (WS_1)	5	3	4
AirTravelBooking (WS_2)	7	4	7

Table 10. Comparison of the number of executions for the proposed and classical algorithms.

QoS constraint structure	Number of executions	
	Classical algorithm	Proposed algorithm
C_1 {QCT of height 0}	1	1
C_1 AND C_2 AND … C_N : {QCT of height 1}	1	1
C_1 OR C_2 OR … C_N : {QCT of height 1}	N	1
(C_1 OR C_2 … C_N) AND (C_1 OR C_2 … C_N) AND…(C_1 OR C_2 … C_N) : {QCT of height 2}	N^2	1
[(C_1 OR C_2 … C_N) AND (C_1 OR C_2 … C_N) AND…(C_1 OR C_2 … C_N)] OR [(C_1 OR C_2 … C_N) AND (C_1 OR C_2 … C_N) AND…(C_1 OR C_2 … C_N)] OR … [(C_1 OR C_2 … C_N) AND (C_1 OR C_2 … C_N) AND… (C_1 OR C_2 … C_N)] : {QCT of height 3)	N^3	1

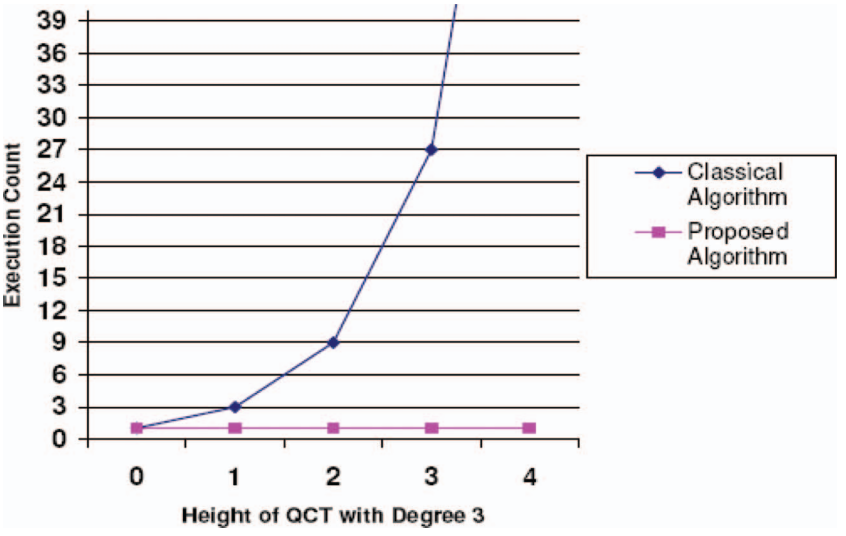

Figure 13. A line graph showing the execution counts for the selection algorithms.

property. Let V_R be the requested value for the QoS property Q_R. The QoS requirement can be expressed as Q_R rp V_R where rp is relational operator. Such a requirement is termed in the QoS constraint structure as a simple QoS constraint. The requirements defined on the multiple QoS properties can be represented as Q_1 rp V_1 AND Q_2 rp V_2 AND,…, Q_N rp V_N which are termed as composite QoS constraints. These two types of requirements select a web service based on the satisfaction of all the simple QoS requirements. Therefore, the QoS requirements involving a single QoS property or multiple QoS properties with AND operators are referred to as *only-if* requirements. The QoS constraint structure also has an ability to express *if-then* type of requirements. For example, the requirement of the form

Q_1 rp V_1 OR Q_2 rp V_2 OR ... Q_N rp V_N is *if-then* requirement as the satisfaction of one of the simple QoS requirements selects the desirable web service. The requirements of this type can be represented in the proposed structure. The QCT structure is extended to represent the requester's alternative QoS constraints which follow the *if-else* structure. Let $C_1, C_2, \ldots, C_N$ be the requirements defined on the QoS. The requirement of the form C_1 XOR C_2 refers to *if-else* structure as the QoS constraint C2 is enforced on the candidate web services if C_1 fails to select a web service. Similarly the structure supports else-if ladder form of requirements i.e. a strong requirement with multiple substitute requirements. For example, the QoS requirement of the form: C_1 XOR C_2 XOR, ..., C_N represents the else-if ladder form of QoS requirement.

The QoS requirements having logical negation, i.e. *not* operator are discouraged in the proposed structure as the logical negation expressions can be represented as negation free form expressions. For example not ($Q_R > V_R$) can be represented as $Q_R \leq V_R$. The proposed QoS constraint structure can be customised for the real-world applications. In the implementation of the QoS broker, the authors have restricted themselves to three alternative QoS constraints ($R = 3$). A separate interface form is activated based on the input value of R. Similarly for each QoS constraint, the degree, i.e. nesting depth, is restricted to 4 ($N = 4$). Thus the QoS constraint structure can be customised depending on the requirements of the web service selection application.

9.3.2. *Real-world application scenario*

Consider e-commerce scenario of book buying in shopping domain as a test case for proof of concepts involved in the selection process. The actors involved in this scenario are the customer, i.e. book buyer, the service providers of various book sellers/distributors, packaging service providers, shipping services and the book seller agent service. The book seller agent provides a system to the buyer with options for his/her purchase and earns money by charging a certain amount on all consumed services. The service providers (book sellers/distributors, packaging services and shipping/cargo services) are providing web services to query their service offerings and to order, i.e. request for the service. Due to the loosely coupled nature of web services, the book seller agent doesn't need to have prior agreements with the service providers. This allows the book seller agent to have access to many services, offering a variety of service options to its buyers. The service providers can offer their services broadly to generate more business for themselves. Figure 14 shows the book buying scenario illustrating the need for the web service selection. In this scenario, the book buyer has some book seller agent software or the buyer agent service which could reside locally on his/her computer.

The conceptual web service architecture facilitates the book sellers, shipping services and packaging service providers to publish their web services into service registry (UDDI). The customer (buying agent) has to query the UDDI registry to gain access to the web services hosted by the service providers. The query execution normally discovers multiple web services providing same functionality. Consider the shipping service: while buying a novel or any book of lesser importance, the customer will not worry about the response time (delivery time) and reliability of the shipping service. In such a situation, the customer randomly picks one shipping service and places the order to ship the book through the web interface. If an urgent

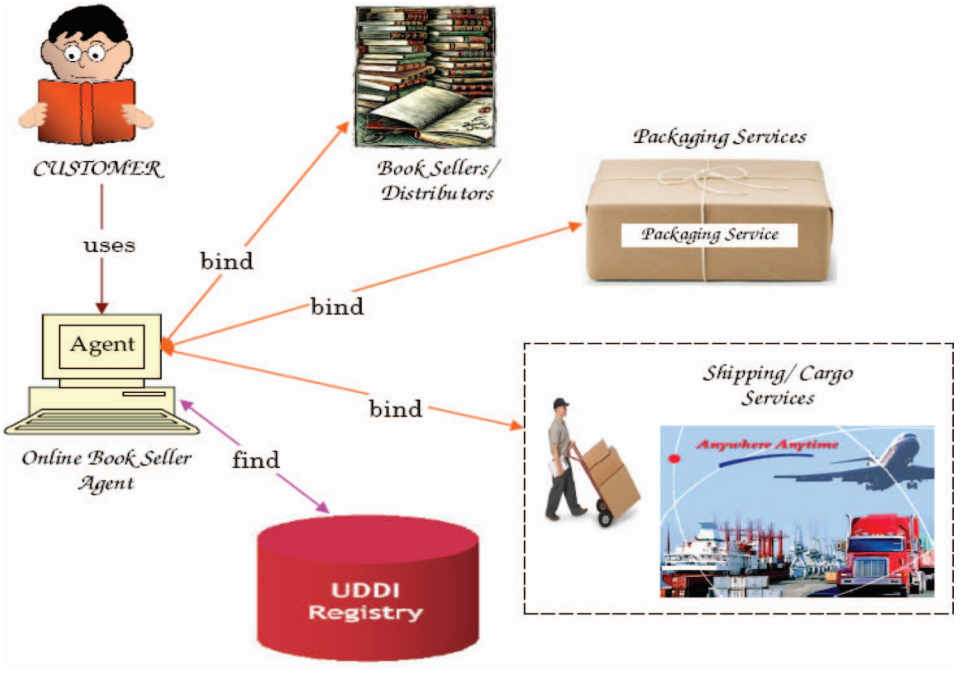

Figure 14. A real-world scenario illustrating the QoS aware web service selection problem.

need arises for the buyer to purchase a costly technical reference manual, then a more trustworthy, speedy and reliable shipping and packaging service is preferred. Similarly, if the customer intends to use many services for his needs then he/she will consider the service price and reputation of each service. In such a scenario, the web service has to be selected such that it satisfies the customer's expectations on the service quality. This necessitates the customer to use tools and techniques to select the best web service based on his QoS requirements. The QoS awareness also improves the competition among service providers to improve their business by providing good service to the customers. The proposed broker based web service architecture for selection is the best architecture to find the desired web service that satisfies the customer's QoS requirements and preferences.

9.3.3. *Alternative implementation of broker based architecture*

The QoS broker based web service architecture can also be implemented using another (alternative to .NET) technology called J2EE 1.4 SDK (Java 2 Platform, Enterprise Edition Standard Development Kit). J2EE platform contains J2SE (Java 2 Standard Edition) SDK, the Sun Java System Application Server and web services APIs to build distributed web or web service applications. The requester can use web browsers like Mozilla or Netscape to establish the communication with web system (J2EE server) through JSP (Java Server Pages) scripts. The JSP scripts dynamically generate HTML pages and also read the information supplied by the provider and requester for web service publishing and selection. The broker can be implemented as a web service on JAX-RPC (Java API for XML-based RPC)

runtime environment. The JAXR (Java API for XML registries) can be used to establish the communication between the broker and the registry i.e. JAXR can be used to discover candidate web services from XML based registries like SAP/IBM/Microsoft UDDI registries. The UDDI registry can also be implemented locally using Java WSDP (Java web Services Developer Pack) registry server. The QoS registry can be implemented as a web service on JAX-RPC runtime environment which in turn can communicate with the MySQL database through Java connector called JDBC (Java Data Base Connectivity).

10. Related work

Quality of service (QoS) is a decisive factor to distinguish functionally similar web services. Recently many researchers have proposed QoS models describing various QoS properties, measurement metrics and verification mechanisms (Menasce 2002, Mani 2002, Yeom *et al*. 2006). Nowadays, researchers are working towards QoS based web service selection to define the selection model/mechanisms (Deora *et al*. 2003, Seo *et al*. 2005). Some attempts have been made to define the selection mechanism for semantic web services (Biligin and Singh 2004, Maximilien and Singh 2004, Wang *et al*. 2006). The QoS based web service selection mechanism involving simple QoS requirements is also proposed in literature (Liu *et al*. 2004, Hu *et al*. 2005, Makripoulias *et al*. 2006). In literature, the web service selection mechanism is implemented using the following three types of extended service oriented architectures. In the first type, the existing UDDI information model is extended for QoS support, i.e. QoS aware service publication and discovery (ShaikhAli *et al*. 2003, Ran 2003, Garcia and Maria 2006). The second type of architecture involves the use of few additional functional components along with UDDI registry to support QoS aware web service publishing and selection (Gao and Wu 2005, Taher *et al*. 2005, Makripoulias *et al*. 2006). The third type of architecture uses the concept of middleware (broker) for QoS aware web service publishing and selection. In broker based architecture, the requester interacts with the broker for dynamic web service selection (Tian *et al*. 2004, Serhani 2005, Tavares and Westphall 2006). The QoS broker is a critical architectural component for the computation and interaction among different roles. The main functionality of the broker is to select an optimum web service for the requester that satisfies his QoS constraints and preferences. The other functionalities of the broker include QoS publishing, QoS verification and certification and QoS management.

Figure 15 shows the evolution of various selection techniques for QoS aware web services. The authors classify QoS based web service selection methods into *two* broad categories as selection for a single task and optimal selection for the composite process. Many researchers have proposed various techniques for the optimal assignment of the web services to the constituent tasks of the composite process/plan (Makris *et al*. 2006, Yu *et al*. 2007, Menasce *et al*. 2008, Canghong *et al*. 2008). In literature, many researchers have proposed various techniques to select the best web service for the specific task based on the QoS. Kerrigan (2006) proposes a selection method which is defined on the single QoS property. There are few efforts towards the QoS aware web service selection based on the QoS requirements defined on the multiple QoS properties (Tian *et al*. 2004, Mou *et al*. 2005). In such selection techniques, the varied preferences of QoS properties are also taken into account to rank the web services for the selection (Liu *et al*. 2004, Hu *et al*. 2005, Guoquan *et al*.

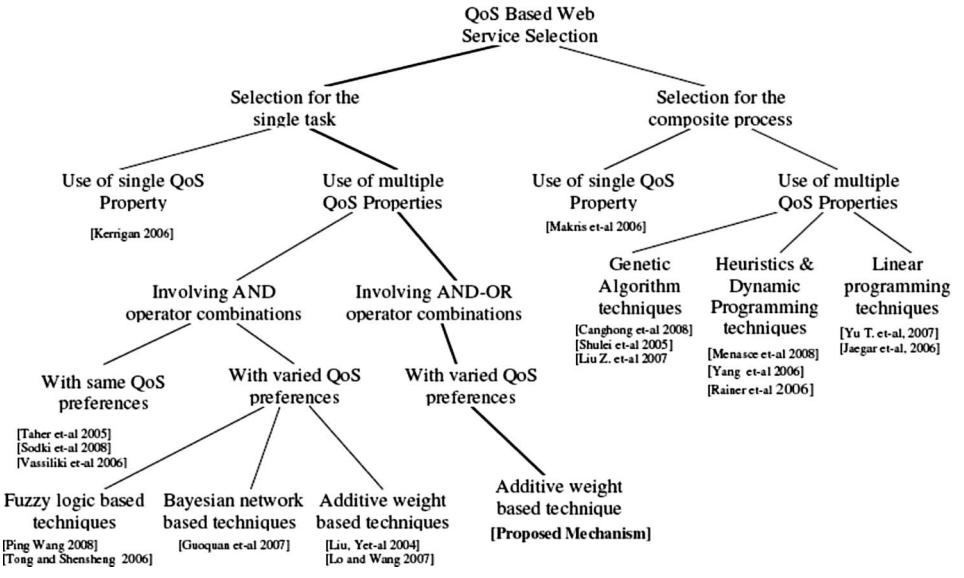

Figure 15. Evolution of QoS aware web service selection methods.

2007, Lo and Wang 2007, Ping Wang 2008). There are few selection mechanisms that do consider the same preferences for all requested QoS properties (Taher *et al*. 2005, Vassiliki *et al*. 2006, Sodki *et al*. 2008). The observation of the selection mechanism evolution tree (Figure 15) reveals the fact that in literature no attempt has been made towards the selection of web services based on requester's QoS constraints involving AND and OR operator combinations having varied QoS property preferences.

Here the authors present the brief description of the selection methods which are closely related to their work. Yetu *et al*. (2004) discuss the issue of quality driven web service selection and propose extended QoS model for web services. The paper presents the design of QoS registry which is responsible for the computation of QoS value for each web service provider. The selection mechanism ranks the web services based on the constraints defined on multiple QoS properties and the rank for a web service is computed as follows. First the QoS property values are normalised individually and then in groups based on usability. The group preference is multiplied by the normalised score to find the final score (rank) of a web service and the web service with highest score refers to the best web service. Taher *et al*. (2005) propose a framework for QoS based dynamic web service selection. The service matchmaking mechanism finds the correlation (Euclidian distance) value between functionally similar web services and the requester's expected QoS and the service with minimum Euclidian distance is selected as a best web service. Hu *et al*. (2005) propose a decision model of QoS criteria called DQoS for evaluating the web services based on multiple QoS properties. The paper models service selection problem as a multiple attribute decision making (MADM) problem which can be solved by using subjective weight mode, single weight mode, objective weight mode and subjective-objective weight mode. Kerrigan (2006) proposes the selection mechanisms for web service execution environment (WSMX) which selects web services based on filtering requirements and ordering preferences defined on single QoS property. In Sodki *et al*. (2008), the web service selection mechanism uses a 2-dimensional Boolean array

called selection matrix as follows. The rows of the matrix represent the web services and the columns represent the QoS properties. The matrix cell value is set to one if the QoS property requirement matches with the advertised value and zero otherwise. The matrix row is selected having maximum number of ones present in it and the corresponding web service becomes the best web service. Lo and Wang (2007) propose CosmosQoS framework to fulfil the requester's QoS requirements. The CosmosQoS defines the web service reputation appraisal model which is composed of three measurement perspectives which are price discrepancy, QoS deviation and historical credibility. The values of these three parameters determine the quality of web service with respect to requester's requirements. Jaeger and Goldmann (2006) define a simple additive weight (SAW) method to rank the functionally similar web services based on local QoS requirements. The classical QoS based service selection techniques proposed in the literature select the web services based on the requester's requirements defined on multiple QoS properties involving only implicit/explicit AND operator combinations with varied preferences.

In this paper, the authors define the important QoS properties of web services to distinguish functionally similar/same web services and categorise them based on the requester's selection point of view as business specific QoS, performance specific QoS and requester's response specific QoS. The paper proposes a structure to represent the variety of requester's constraints defined on the multiple QoS properties involving AND and OR operators with varied preferences in a structured way. The authors propose a tree structure called quality constraint tree (QCT) to represent the requester's QoS constraints and preferences. The authors provide the mechanism to transform the QCT of a given QoS constraint into an XML form which can be easily embedded into SOAP message for the program to program selection mechanism. The authors extend the QoS constraint structure to specify the requester's alternative QoS constraints for the web service selection and define an extended QCT to model such QoS constraint substitutes. The authors propose a novel web service selection mechanism which selects the eligible web services from functionally similar web services by filtering the low quality web services using QoS constraints having multiple QoS properties involving both AND and OR operators. The selected candidate web services are then ranked based on the QoS property values of web services and the requester's QoS constraint preferences. To realise the web service selection process, the authors propose the QoS broker based web service architecture for selection which facilitates the QoS aware service providers to publish their web services. The architecture allows the requesters to specify their QoS constraints and preferences for the web service selection. The architecture also manages QoS properties of various web services by obtaining an authentic response on the service consumption from the legitimate requesters of web services. The proposed broker based architecture and the selection mechanism also effectively selects and ranks web services based on the requester's alternative QoS constraints defined on the multiple QoS properties involving AND and OR operators.

11. Conclusion and future work

The web service selection is one of the major problems because the requester normally expects certain non-functional properties of business driven web services to

be satisfied by the providers. The best selection made by one requester may not be a good selection for the other requester as the quality requirements normally vary from one requester to the other. In this paper, the authors propose QoS vocabulary for business driven web service publishing and selection which categorises QoS properties of web services based on requester's selection point of view as business specific QoS, performance specific QoS and requester's response specific QoS. The paper explores different types of requester's QoS constraints and suggests a tree model called quality constraint tree (QCT) and its XML equivalent to represent requester's QoS requirements defined on multiple QoS properties involving AND and OR operators with varied preferences. The authors propose the QoS broker based architecture for QoS-aware web service publishing and selection. The QoS broker employs a selection mechanism which finds the best web service for the requester based on his QoS constraints and preferences.

The paper also explores a method to resolve the tie (web service with same computed score) that may occur among qualitatively competitive web services during the selection process. The authors define a few web service provider qualities like provider popularity, provider size and provider existence to distinguish qualitatively competitive web services. The paper proposes a model to represent the requester's set of alternative QoS requirements by extending QCT structure. The web service selection algorithm is also extended to handle requester's set of alternative QoS constraints having diminishing preferences.

As a future work, the broker based architecture can be extended to incorporate various business and service offers of web service providers. The service offers like service charge discounts, gifts after service consumption, free service executions etc. can be used to discriminate functionally similar web services. The web service selection mechanism can be enriched to handle the requester's demands on the service offers to find the most profitable web service. The information related to the requester's QoS behaviour can also be used to discriminate and rank the functionally similar and qualitatively competitive web services.

To realise the QoS broker based web service publishing and selection, there is a need for the mechanism to estimate the performance specific QoS properties of web services. The architecture can also be augmented with an additional trusted third party role called QoS Certifier to verify and certify the performance QoS properties of web services for the legitimacy. A mechanism is required at the broker site, to authenticate the requesters and web service providers who interact with the broker, to avoid the fake responses and registration of dummy web services from them. To estimate the requester's response specific QoS properties, there is a need for the response format which can include all necessary details to be furnished by the service requesters. As the requester's behaviour on QoS and the service usage context both decide the response ratings, a scheme is needed to capture the requester's behaviour on QoS and the service usage context. Since the QoS broker itself is implemented as a web service, it has to be published in UDDI registry with all adequate details including the service publishing format, query formats and the portal link. The portal of QoS broker can contain detailed information related to broker features, usage tips, regulations for the requester/provider registration and the formats for service publishing and selection.

References

Ali, A.S., Ludwig, S.A., and Rana, O.F., 2005. A Cognitive Trust-based Approach for web Service Discovery and Selection. *In*: *Proceedings of the 3rd European conference on web services (ECOWS'05)*. IEEE Computer Society, 38–49.

Blake, W.T. and Wagner, M., 2003. Towards personalised selection of web services. *In*: *Proceedings of the 12th international WWW conference*, ACM.

Bilgin, A.S. and Singh, M.P., 2004. A DAML-based repository for QoS-aware web service selection. *In*: *Proceedings of the IEEE international conference on web services (ICWS'04)*. IEEE Computer Society.

Canghong, J., MinghuiWu, Tao, J., and Jing, Y., 2008. Combine automatic and manual process on web service selection and composition to support QoS. *In*: *Proceedings of the 12th international conference on computer supported cooperative work in design (CSCWD 2008)*. IEEE Computer Society.

Chung, M., *et al*., 2005. Improved matching algorithm for services described by OWL-S. *In*: *Proceedings of the ICACT 2006*, 1510–1513.

Day, J. and Ralph, D., 2004. Selecting the best web service. *In*: *Proceedings of the 2004 conference of the centre for advanced studies on collaborative research*, IBM Press.

Deora, V., *et al*., 2003. A quality of service management framework based on user expectations. *In*: *Proceedings of the ICSOC 2003*, LNCS 2910, 104–114.

D'Mello, D.A. and Ananthanarayana, V.S., 2008a. A quality of service (QoS) model for web services. *In*: *Proceedings of the international conference on emerging technologies and applications in engineering, technology, and sciences (ICETAETS 2008)*, 14–15 January 2008, Saurashtra University, Gujrat, Vol. 1, 832–837.

D'Mello, D.A. and Ananthanarayana, V.S., 2008b. A QoS broker based architecture for dynamic web service selection. *In*: *Proceedings of the 2nd Asia international conference on modelling and simulation (AMS)*, IEEE Computer Society, 101–106.

Gao, Z. and Wu, G., 2005. Combining QoS-based service selection with performance prediction. *In*: *Proceedings of the IEEE international conference on e-business engineering (ICEBE'05)*, IEEE Computer Society.

Garcia, D.Z.G. and Maria, B.F.T., 2006. A web service architecture providing QoS management. *In*: *Proceedings of the 4th Latin American web congress (LA-WEB'06)*, IEEE Computer Society.

Guoquan, W., *et al*., 2007. A Bayesian network based QoS assessment model for web services. *In*: *Proceedings of the IEEE international conference on services computing (SCC 2007)*, IEEE Computer Society.

Hu, J., *et al*., 2005. Quality driven web services selection. *In*: *Proceedings of the IEEE international conference on e-business engineering (ICEBE'05)*, IEEE Computer Society.

Jaeger, M., *et al*., 2005. Ranked matching for service descriptions using OWL-S. *Kommunikation in Verteilten Systemen (KiVS)*, Teil II, Springer, 91–102.

Jaeger, M. and Goldmann, G.R., 2006. SENECA – simulation of algorithms for the selection of web services for compositions. *In*: *Proceedings of TES 2005*, LNCS 3811, 84–97.

Kerrigan, M., 2006. Web service selection mechanisms in the Web service execution environment (WSMX). *In*: *Proceedings of the 21st annual ACM symposium on applied computing (SAC'06)*, ACM, 1664–1668.

Kreger, H., 2001. *Web services conceptual architecture (WSCA 1.0)* [online]. IBM Software Group. Available from: http://www.ibm.com/software/solutions/webservices/pdf/wsca.pdf [Accessed 4 April 2007].

Liu, Y., Ngu, A.H.H., and Zeng, L., 2004. QoS computation and policing in dynamic web service selection. *In*: *Proceedings of the WWW 2004*, ACM, 66–73.

Liu, Z., *et al*., 2007. Solving fuzzy QoS constraint satisfaction technique for web service selection. *In*: *Proceedings of the international conference on computational intelligence and security workshops*, IEEE Computer Society.

Lo, N.W. and Wang, C., 2007. Web services QoS evaluation and service selection framework – a proxy-oriented approach. *In*: *Proceedings of the 2007 IEEE region 10 conference (TENCON 2007)*, IEEE Computer Society.

Makripoulias, Y., *et al*., 2006. Web service discovery based on quality of service. *In*: *Proceedings of the IEEE international conference on computer systems and applications*, IEEE Computer Society.

Makris, C., *et al*., 2006. Efficient and adaptive discovery techniques of web services handling large data sets. *Journal of Systems and Software*, 79 (4), 480–495.

Mani, A. and Nagarajan, A., 2002. Understanding quality of service for web services [online] IBM Developer Works. Available from: www.ibm.com/developerworks/library/ws-quality.html [Accessed 16 May 2007].

Marchim, M., Mileo, A., and Provetti, A., 2005. Declarative policies for web service selection. *In*: *Proceedings of the IEEE international workshop on policies for distributed systems and networks (POLICY'05)*, IEEE Computer Society.

Martin, D., *et al*., 2005. Bringing semantics to web services: The OWL-S approach. *In*: *Proceedings of the SWSWPC 2004*, Springer LNCS 3387, 26–42.

Maximilien, E.M. and Singh, M.P., 2004. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8 (5), 84–93.

Menasce, D.A., 2002. QoS issues in web services. *IEEE Internet Computing*, 6 (6), 72–75.

Menasce, D.A., Casalicchio, E., and Dubey, V., 2008. A heuristic approach to optimal service selection in service oriented architectures. *In*: *Proceedings of the 7th international workshop on software and performance (VOSP'08)*, Princeton, NJ, USA, 13–23.

Mou, Y., *et al*., 2005. Interactive web service choice-making based on extended QoS model. *In*: *Proceedings of the 2005 the 5th international conference on computer and information technology (CIT'05)*, IEEE Computer Society.

Ping, W., 2008. QoS-aware web services selection with intuitionistic fuzzy set under consumer's vague perception. *Journal of Expert Systems with Applications*, 36 (3), 4460–4466.

Rainer, B., *et al*., 2006. Heuristics for QoS-aware web service composition. *In*: *Proceedings of the IEEE international conference on web services (ICWS'06)*, IEEE Computer Society, 72–82.

Ran, S., 2003. A model for web services discovery with QoS. *SIGecom Exchange*, 4 (1), 1–10.

Riegen, C.V. (ed.) 2002. *UDDI Version 2.03 Data Structure Reference* [online]. OASIS Open 2002–2003. Available from: http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm [Accessed 8 November 2007].

Rocco, D. and Caverlee, J., 2005. Domain-specific web service discovery with service class descriptions. *In*: *Proceedings of the IEEE international conference on web services (ICWS'05)*, IEEE Computer Society.

Seo, Y.J., Jeong, H.Y., and Song, Y.J., 2005. Best web service selection based on the decision making between QoS criteria of service. *In*: *Proceedings of the ICESS 2005,* LNCS 3820, 208–419.

Serhani, M.A., *et al*., 2005. A QoS broker-based architecture for efficient web service selection. *In*: *Proceedings of the IEEE international conference on web services (ICWS'05)*, IEEE Computer Society.

ShaihkAki, A., *et al*., 2003. UDDIe: An extended registry for web services. *In*: *Proceedings of the symposium on applications and the internet workshops*IEEE Computer society, 85–89.

Shen, Z. and Su, J., 2005. Web service discovery based on behavior signatures. *In*: *Proceedings of the IEEE international conference on services computing (ICSC 2005)*, IEEE Computer Society.

Shulei, L., *et al*., 2005. A dynamic web service selection strategy with QoS global optimisation based on multi-objective genetic algorithm. *In*: *Proceedings of the GCC 2005*, LNCS 3795, 84–89.

Sodki, C., Youakim, B., and Frédérique, B., 2008. Enhancing web service selection by QoS-based ontology and WS-Policy. *In*: *Proceedings of the 2008 ACM symposium on applied computing (SAC'08)*, 16–20 March, Fortaleza, Ceará, Brazil, 2426–2431.

Taher, L., Khatib, H.E., and Basha, R., 2005. A framework and QoS matchmaking algorithm for dynamic web service selection. *In*: *Proceedings of the 2nd international conference on innovations in information technology (IIT'05)*, 1–10.

Tavares, R.K. and Westphall, C.B., 2006. An architecture to provide QoS in web services. *In*: *Proceedings of IEEE ICC 2006*, IEEE Computer Society.

Tian, M., *et al*., 2004. Efficient selection and monitoring of QoS-aware web services with the WS-QoS framework. *In*: *Proceedings of the IEEE/WIC/ACM international conference on web intelligence (WI'04)*, IEEE Computer Society.

Tong, H. and Shensheng, Z., 2006. A fuzzy multi-attribute decision making algorithm for web services selection based on QoS. *In*: *Proceedings of the 2006 IEEE Asia-Pacific conference on services computing (APSCC'06)*, IEEE Computer Society.

Vassiliki, D., *et al*., 2006. Techniques to support web service selection and consumption with QoS characteristics. *Journal of Network and Computer Applications*, 31, 108–130.

Wang, X., *et al*., 2006. A QoS-aware selection model for semantic web services. *In*: *Proceedings of ICSOC 2006*, LNCS 4294, 390–401.

Wang, Y. and Stroulia, E., 2003. Flexible interface matching for web-service discovery. *In*: *Proceedings of the 4th international conference on web information systems engineering (WISE'03)*, IEEE Computer Society.

Wu, J. and Wu, Z., 2005. Similarity-based web service matchmaking. *In*: *Proceedings of the IEEE international conference on services computing (SCC'05)*, IEEE Computer Society.

Yang, L., *et al*., 2006. A dynamic web service composite platform based on QoS of services. *In*: *Proceedings of APweb workshops 2006*, LNCS 3842, 709–716.

Yeom, G., Yun, T., and Min, D., 2006. A QoS model and testing mechanism for quality-driven web services selection. *In*: *Proceedings of the 2nd international workshop on collaborative computing, integration, and assurance (SES–WCCIA'06)*, IEEE Computer Society.

Yu, T., Zhang, Y., and Lin, K., 2007. Efficient algorithms for web services selection with end-to-end QoS Constraints. *ACM Transactions on the Web*, 1 (1), 1–26.

Zhuang, L. and YuanFei, H., 2006. Matchmaking for semantic web services based on OWL-S. *In*: *Proceedings of the 1st international conference on semantics, knowledge, and grid (SKG 2005)*, IEEE Computer Society.