QoS Computation and Policing in Dynamic Web Service Selection

Yutu Liu [*]
Department of Computer Science
Texas State University
San Marcos, Texas
alinux@neo.tamu.edu

Anne H.H. Ngu
Department of Computer Science
Texas State University
San Marcos, Texas
hn12@txstate.edu

Liangzhao Zeng
IBM T.J. Watson Research Center
New York, USA
lzeng@us.ibm.com

ABSTRACT

The emerging Service-Oriented Computing (SOC) paradigm promises to enable businesses and organizations to collaborate in an unprecedented way by means of standard web services. To support rapid and dynamic composition of services in this paradigm, web services that meet requesters' functional requirements must be able to be located and bounded dynamically from a large and constantly changing number of service providers based on their Quality of Service (QoS). In order to enable quality-driven web service selection, we need an open, fair, dynamic and secure framework to evaluate the QoS of a vast number of web services. The fair computation and enforcing of QoS of web services should have minimal overhead but yet able to achieve sufficient trust by both service requesters and providers. In this paper, we presented our open, fair and dynamic QoS computation model for web services selection through implementation of and experimentation with a QoS registry in a hypothetical phone service provisioning market place application.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Management, Experimentation, Algorithms

Keywords

QoS, Web Services, extensible QoS Model, Ranking of QoS

1. INTRODUCTION

Web services are self-describing software applications that can be advertised, located, and used across the Internet using a set of standards such as SOAP, WSDL, and UDDI [8]. Web services encapsulate application functionality and information resources, and make them available through standard programmatic interfaces. Web services are viewed as one of the promising technologies that could help business entities to automate their operations on the web on a large scale by automatic discovery and consumption of services. Business-to-Business(B2B) integration can be achieved on a demand basis by aggregating multiple services from different providers into a value-added composite service.

In the last two years, the process-based approach to web service composition has gained considerable momentum and standardization [1]. However, with the ever increasing number of functional similar web services being made available on the Internet, there is a need to be able to distinguish them using a set of well-defined Quality of Service (QoS) criteria. A service composition system that can leverage, aggregate and make use of individual component's QoS information to derive the optimal QoS of the composite service is still an ongoing research problem. This is partly due to the lack of an extensible QoS model and a reliable mechanism to compute and police QoS that is fair and transparent to both service requesters and providers.

Currently, most approaches that deal with QoS of web services only address some generic dimensions such as price, execution duration, availability and reliability [12, 6]. In some domains, such generic criteria might not be sufficient. QoS model should also include domain specific criteria and be extensible. Moreover, most of the current approaches rely on service providers to advertise their QoS information or provide an interface to access the QoS values, which is subject to manipulation by the providers. Obviously, service providers may not advertise their QoS information in a "neutral" manner, for example, execution duration, reliability, etc. In approaches where QoS values are solely collected through active monitoring, there is a high overhead since QoS must be checked constantly for a large number of web services. On the other hand, an approach that relies on a third party to rate or endorse a particular service provider is expensive and static in nature. In our framework, the QoS model is extensible, and QoS information can either be provided by providers, computed based on execution monitoring by the users, or collected via requesters feedback, depending on the characteristics of each QoS criterion. In a nutshell, we propose a framework that aims at advancing the current state of the art in QoS modeling, computation and policing. There are three key aspects to our work:

- **Extensible QoS model**. In the presence of multiple web services with overlapping or identical functionality, service requesters need objective QoS criteria to

[*]Yutu Liu is currently pursuing his graduate study at Department of Computer Science, Texas A&M. This work was conducted while he was a master student at Texas State University

distinguish one service from another. We argue that it is not practical to come up with a standard QoS model that can be used for all web services in all domains. This is because QoS is a broad concept that can encompass a number of context-dependent non-functional properties such as privacy, reputation and usability. Moreover, when evaluating QoS of web services, we should also take into consideration domain specific criteria. For example, in the domain of phone service provisioning, the penalty rate for early termination of a contract and compensation for non-service, offered in the service level agreement are important QoS criteria in that domain. Therefore, we propose an extensible QoS model that includes both the generic and domain specific criteria. In our approach, new domain specific criteria can be added and used to evaluate the QoS of web services without changing the underlying computation model.

- **Preference-oriented service ranking**. Different users may have different preferences or requirements on QoS. It is important to be able to represent QoS from the perspective of service requesters' preference. For example, service selection may be driven completely by price, regardless of the time it takes to execute the service. A different requester may be very service sensitive. This means that criteria such as penalty rate or the ability to return the goods after the purchase are viewed as more important than the price and the time. Another service selection may be driven completely by time because of tight deadlines. A QoS model should provide means for users to accurately express their preferences without resorting to complex coding of user profiles.

- **Fair and open QoS computation**. Once a set of QoS criteria have been defined for a particular domain, we must ensure that QoS information is collected in a fair manner. In our framework, QoS information can be collected from service properties that are published by providers, execution monitoring, and requesters' feedback based on the characteristic of quality criterion. For example, execution price can be provided by service providers, execution duration can be computed based on service invocation instances, while service reputation is based on service requesters' feedback. We also build a policing mechanism that will prevent the manipulation of QoS value from a single service requester by requiring each requester to have a valid pair of user id and password to update the QoS registry. Furthermore, this pair of id and password must be verified by the service providers at the consumption of the service to ensure that only the actual consumer of the service is allowed to give feedback. On the other hand, to be completely fair, providers can review those criteria and can improve their QoS if they desire to. Moreover, providers can update their QoS information (e.g., execution price, penalty rate) at any time. Providers can also check the QoS registry to see how their QoS is ranked among other service providers.

We believe that an extensible, transparent, open and fair QoS model is necessary for the selection of web services. Such a model can benefit all participants. Although this model requires all requesters to update their usage experiences with a particular type of service in a registry, this overhead on the user is not large. In return, QoS for a particular service is actively being policed by all requesters. Each requester can search the registry to get the most-updated QoS of listed providers. Service providers can view and change their services at any time. With such an open and dynamic definition of QoS, a provider can operate its web services to give its end-users the best user experience.

This paper is organized as follows. In Section 2, we give the details of an extensible QoS model and its computation. In Section 3, we describe the implementation of QoS registry and explain how QoS information can be collected based on active monitoring and active user feedback. Section 4 discusses the experiments that we conducted to confirm the fairness and the validity of the various parameters used in QoS computation. Finally, we discuss related work in Section 5 and conclude in Section 6.

2. QOS-BASED SERVICE SELECTION

Currently, in most SOC frameworks, there is a service broker which is responsible for the brokering of functional properties of web services. In our framework, we extend capability of the service broker to support non-functional properties. This is done by introducing an extensible and multiple dimensions QoS model in the service broker. We aim to evaluate QoS of web services using an open, fair and transparent mechanism. In the following subsections, we first introduce the extensible QoS model, and then we give the details on how to evaluate web service based on our model.

2.1 Extensible QoS Model

In this section, we propose an extensible QoS model, which includes generic and domain or business specific criteria. The generic criteria are applicable to all web services, for example, their pricing and execution duration. Although the number of QoS criteria discussed in this paper is limited (for the sake of illustration), our model is extensible. New criteria (either generic or domain specific) can be added without fundamentally altering the underlying computation mechanism as shown in Section 2.2. In particular, it is possible to extend the quality model to integrate non-functional service characteristics such as those proposed in [7], or to integrate service QoS metrics such as those proposed by [11].

2.1.1 Generic quality criteria

We consider three generic quality criteria which can be measured objectively for elementary services: (1) *execution price*, (2)*execution duration*, and (3) *reputation*. Criteria such as availability and reliability is not required in our model due to the use of active user feedback and execution monitoring.

- **Execution price**. This is the amount of money which a service requester has to pay to the service provider to use a web service such as checking a credit, or the amount of money the service requester has to pay to the service provider to get a commodity like an entertainment ticket or a monthly phone service. Web service providers either directly advertise the execution price of their services, or they provide means for potential requesters to inquire about it. Let s be one web service, then $q_{pr}(s)$ is the execution price for using that service s.

- **Execution duration.** The execution duration $q_{du}(s)$ measures the expected delay in seconds between the moment when a request is sent and the moment when the service is rendered. The execution duration is computed using the expression $q_{du}(s) = T_{process}(s) + T_{trans}(s)$, meaning that the execution duration is the sum of the processing time $T_{process}(s)$ and the transmission time $T_{trans}(s)$. Execution time is obtained via active monitoring.

- **Reputation.** The reputation $q_{rep}(s)$ of a service s is a measure of its trustworthiness. It mainly depends on end user's experiences of using the service s. Different end users may have different opinions on the same service. The value of the reputation is defined as the average ranking given to the service by end users, i.e., $q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$, where R_i is the end user's ranking on a service's reputation, n is the number of times the service has been graded. Usually, end users are given a range to rank Web services. For example, in Amazon.com, the range is $[0, 5]$.

2.1.2 Business related criteria

The number of business related criteria can vary in different domains. For example, in phone service provisioning domain, the penalty rate for the early termination and the fixed monthly charge are important factors for users to consider when selecting a particular service provider. We use the generic term **usability** to group all business related criteria. In our chosen application, we measure usability from three aspects, transaction, compensation rate and the penalty rate.

- **Transaction** support is used for maintaining data consistency. In prior QoS models, no transactional criteria is being used in the computation of QoS value. However, from the perspective of a requester, whether a web service provides an undo procedure to rollback the service execution in certain period without any charges is an important factor that will affect his/her choice. Transactional property can be evaluated by two dimensions: whether undo procedure is supported $q_{tx}(s)$ and what's the time constraints $q_{cons}(s)$ on undo procedure. It should be noted that $q_{tx}(s) = 0/1$, where 1 indicate the web service supports transaction and 0 otherwise; $q_{cons}(s)$ indicates the duration of undo procedure is allowed.

- **Compensation rate** $q_{comp}(s)$ of a web service indicates percentage of the original execution price that will be refunded when the service provider can not honor the committed service or deliver the ordered commodity.

- **Penalty rate** $q_{pen}(s)$ of a web service indicates what percentage of the original price service requesters need to pay to the provider when he/she wants to cancel the committed service or ordered commodity after the time out period for transaction to roll back is expired.

2.2 Web Service QoS Computation

The QoS registry is responsible for the computation of QoS value for each service provider. Assuming that there is a set of web services that have the same functional properties, where S ($S = \{s_1, s_2, ..., s_n\}$), web service computation algorithm determines which service s_i is selected based on end user's constraints. Using m criteria to evaluate web service, we can obtain the following matrix $\mathbf{Q}$. Each row in $\mathbf{Q}$ represents a web service s_i, while each column represents one of the QoS criteria.

$$\mathbf{Q} = \begin{pmatrix} q_{1,1} & q_{1,2} & \dots & q_{1,m} \\ q_{2,1} & q_{2,2} & \dots & q_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ q_{n,1} & q_{n,2} & \dots & q_{n,m} \end{pmatrix} \quad (1)$$

In order to rank the web services, the matrix $\mathbf{Q}$ need to be normalized. The purposes of normalization are: 1) to allow for a uniform measurement of service qualities independent of units. 2) to provide a uniform index to represent service qualities for each provider. Provider can increase and decrease his/her quality index by entering a few parameters, 3) to allow setting a threshold regarding the qualities. The number of normalizations performed depends on how the quality criteria are grouped. In our example criteria given in Section 2, we need to apply two phases of normalization before we can compute the final **QoS** value. The second normalization is used to provide uniform representation of a group of quality criteria (e.g. usuability) and set threshold to a group of quality criteria.

- **First Normalization**
Before normalizing matrix $\mathbf{Q}$, we need to define two arrays. The first array is $N = \{n_1, n_2, ..., n_m\}$ with $1 \le j \le m$. The value of n_j can be 0 or 1. $n_j = 1$ is for the case where the increase of $q_{i,j}$ benefits the service requester while $n_j = 0$ is for the case where the decrease of $q_{i,j}$ benefits the service requester. The second array is $C = \{c_1, c_2, ..., c_m\}$. Here c_j is a constant which sets the maximum normalized value. Each element in matrix $\mathbf{Q}$ will be normalized using the following equation 2 and equation 3.

$$v_{i,j} = \begin{cases} \frac{q_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}} & \text{if } \frac{1}{n}\sum_{i=1}^{n} q_{i,j} \neq 0 \\ & \text{and } \frac{q_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}} < c_j \\ & \text{and } \; n_j = 1 \\ c_j & \text{if } \frac{1}{n}\sum_{i=1}^{n} q_{i,j} = 0 \\ & \text{and } \; n_j = 1 \\ & \text{or } \frac{q_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}} \ge c_j \end{cases} \quad (2)$$

$$v_{i,j} = \begin{cases} \frac{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}}{q_{i,j}} & \text{if } q_{i,j} \neq 0 \\ & \text{and } \; n_j = 0 \\ & \text{and } \frac{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}}{q_{i,j}} < c_j \\ c_j & \text{if } q_{i,j} = 0 \\ & \text{and } \; n_j = 0 \\ & \text{or } \frac{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}}{q_{i,j}} \ge c_j \end{cases} \quad (3)$$

In the above equations, $\frac{1}{n}\sum_{i=1}^{n} q_{i,j}$ is the average value of quality criteria **j** in matrix **Q**. Applying these two equations to **Q**, we get matrix $\mathbf{Q}^{'}$ which is shown below:

$$\mathbf{Q}^{'} = \begin{pmatrix} v_{1,1} & v_{1,2} & \dots & v_{1,m} \\ v_{2,1} & v_{2,2} & \dots & v_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ v_{n,1} & v_{n,2} & \dots & v_{n,m} \end{pmatrix} \quad (4)$$

Example 1. Data in the following example is taken from our QoS registry implementation. We assume that there are two web services in S and that their values of quality criteria are: $\mathbf{Q}$=($q_{1,1}$, $q_{1,2}$, $q_{1,3}$, $q_{1,4}$, $q_{1,5}$, $q_{1,6}$, $q_{1,7}$,$q_{2,1}$, $q_{2,2}$, $q_{2,3}$, $q_{2,4}$, $q_{2,5}$, $q_{2,6}$, $q_{2,7}$) = (25,1,60,0.5,0.5,100,2.0,40,1,200,0.8,0.1,40,2.5). The quality criteria are in the order of Price, Transaction, Time Out, Compensation Rate, Penalty Rate, Execution Duration and Reputation. For array **N**, since the increase of price, penalty rate and execution duration doesn't benefit the service requester, and the increase of the remaining quality criteria benefits the service requester, the value for each element in array **N** is $\{0, 1, 1, 1, 0, 0, 1\}$. For array **C**, each element in the array is set to 5 which is considered as a reasonable maximum normalized value in this implemented domain. Using equation 2 and equation 3, we have the normalized matrix: $\mathbf{Q}^{'}$=($v_{1,1}$, $v_{1,2}$, $v_{1,3}$, $v_{1,4}$, $v_{1,5}$, $v_{1,6}$, $v_{1,7}$,$v_{2,1}$, $v_{2,2}$, $v_{2,3}$, $v_{2,4}$, $v_{2,5}$, $v_{2,6}$, $v_{2,7}$) = (1.3, 1.0, 0.462, 0.769, 0.64, 0.7, 0.8894, 0.8134, 1.0, 1.538, 1.23, 3.0, 1.75, 1.111)

- **Second Normalization**
In our QoS model, quality criterion can also be represented as a group and manipulated as a group. For example, the usability criteriion. Each group can contain multiple criteria. For example, both compensation and penalty rates belong to the usability group. To compute the final QoS value for each web service, we introduce Matrix **D** and Matrix **G**. Matrix **D** is used to define the relationship between quality criteria and quality groups. Each row in Matrix **D** represents a quality criterion, and each column in Matrix **D** represents one quality group value. Matrix **G** represents QoS information based on the values of quality group of web services. Each row in Matrix **G** represents a web service, and each column in Matrix **G** represents one quality group value. Matrix **D** and Matrix **G** are shown below:

$$\mathbf{D} = \begin{pmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,l} \\ d_{2,1} & d_{2,2} & \dots & d_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m,1} & d_{m,2} & \dots & d_{m,l} \end{pmatrix} \quad (5)$$

$$\mathbf{G} = \begin{pmatrix} g_{1,1} & g_{1,2} & \dots & g_{1,l} \\ g_{2,1} & g_{2,2} & \dots & g_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ g_{n,1} & g_{n,2} & \dots & g_{n,l} \end{pmatrix} \quad (6)$$

Here, l is the total number of groups of quality criteria. For the value of each element in matrix **D**, $d_{i,j} = 1$ if the **ith** quality criterion in $\mathbf{Q}^{'}$ is included in **jth** group in **G**. By applying Matrix **D** to $\mathbf{Q}^{'}$, we have matrix **G**. The equation is shown below:

$$G = Q^{'} * D \quad (7)$$

To normalize matrix **G**, two arrays are needed. In the first array $T = \{t_1, t_2, ..., t_l\}$, t_j is a constant which sets the maximum normalized value for the group **j**. In the second array $F = \{f_1, f_2, ..., f_l\}$, f_j is a weight for group **j** such as price sensitivity and service sensitivity etc. This is used to express users' preferences over **jth** group. Each element in matrix **G** will be normalized using equation 8.

$$h_{i,j} = \begin{cases} \frac{g_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} g_{i,j}} & \text{if } \frac{1}{n}\sum_{i=1}^{n} g_{i,j} \neq 0 \\ & \text{and } \frac{g_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} g_{i,j}} < t_j \\ t_j & \text{if } \frac{1}{n}\sum_{i=1}^{n} g_{i,j} = 0 \\ & \text{or } \frac{g_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} g_{i,j}} \geq t_j. \end{cases} \quad (8)$$

In the above equations, $\frac{1}{n}\sum_{i=1}^{n} g_{i,j}$ is the average value of group criteria **j** in matrix **G**. Applying equation 8 to **G**, we get matrix $\mathbf{G}^{'}$ which is shown below:

$$\mathbf{G}^{'} = \begin{pmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,l} \\ h_{2,1} & h_{2,2} & \dots & h_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ h_{n,1} & h_{n,2} & \dots & h_{n,l} \end{pmatrix} \quad (9)$$

Finally, we can compute the QoS value for each web service by applying array **F** to matrix $\mathbf{G}^{'}$. The formula of QoS is shown below:

$$QoS(s_i) = \sum_{j=1}^{l} (h_{i,j} * f_j) \quad (10)$$

Example 2. This example continues the computation of QoS from Example 1. For array **T**, each element in the array is set to 5 which is considered as a reasonable maximum normalized value in this implemented domain. For array **F**, each weight is given as 1. This means user gives the same preference to every group. The defined **D**, which determines the grouping of the quality criteria in this example, is shown below:

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The column in **D** is in the order of Price, Time, Usability and Reputation. First, by applying **D** to matrix $\mathbf{Q}'$, we obtained $\mathbf{G}$= (0.813, 1.75, 7.068, 1.111, 1.3, 0.7, 3.531, 0.889). Second, we apply equation 8 to $\mathbf{G}$ to have a normalized $\mathbf{G}'$= ($h_{1,1}$, $h_{1,2}$, $h_{1,3}$, $h_{1,4}$, $h_{2,1}$, $h_{2,2}$, $h_{2,3}$, $h_{2,4}$) = (0.769, 1.429, 1.334, 1.111, 0.946, 0.571, 0.666, 0.889). Finally, using equation 10, we have $\mathbf{QoS(s_1)}$=4.643 and $\mathbf{QoS(s_2)}$=3.072

3. IMPLEMENTATION OF QOS REGISTRY

To demonstrate our proposed QoS model, we implemented a QoS registry as shown in Figure 1 within a hypothetical phone service (UPS) provisioning market place. The UPS market place is implemented using BEA Weblogic Workshop web service toolkit. It consists of various service providers who can register to provide various type of phone services such as long distance, local phone service, wireless and broadband. The market place also has a few credit checking agencies which offer credit checking service for a fee. The UPS market place has web interfaces which allow a customer to login and search for phone services based on his/her preferences. For example, the customer can specify whether the search for a particular type of service should be price or service sensitive. A price sensitive search will return a list of service providers who offers the lowest price. A service sensitive search will return a list of service providers with the best rated services.

The market place also has web interfaces that allow service providers to register their web services with the QoS registry, update their existing web services in the registry or view their QoS ranking in the market place. The registry's other interfaces are available for requesters/end-users to give feedback on the QoS of the web services which they just consumed.

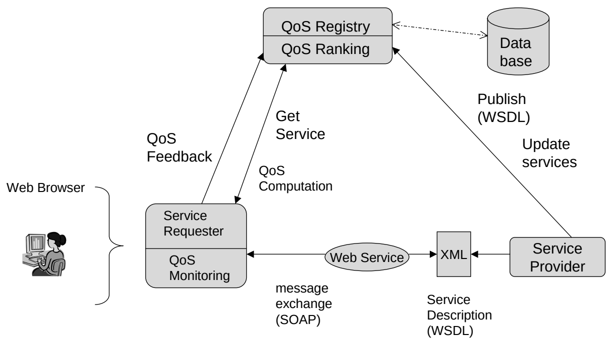

Figure 1: *Architecture Diagram of QoS Registry.*

3.1 Collecting Service Quality Information

In our framework, we distinguish two types of criterion: *deterministic* and *non-deterministic*. Here, deterministic indicates that the value of QoS quality criterion is known or certain when a service is invoked, for example, the execution price and the penalty rate. The non-deterministic is for QoS quality criterion that is uncertain when web service is invoked, for example execution duration. For deterministic criterion, we assume that service providers have mechanisms to advertise those values through means such as web service quality-based XML language as described in [2]. How all service providers come to an agreement of deterministic set of QoS criteria to advertise is beyond the scope of this paper. In this section, we discuss how all QoS criteria in our model can be collected in a fair, open and objective manner via active monitoring and active user's feedback.

- **Collecting quality information from active execution monitoring.** The actual service execution duration is collected by the service requester. This requires interfaces used by all service requesters to implement some mechanisms to log the actual execution time. Although this approach puts more burden on the service requester, it has the following advantages over approaches where the service broker or QoS registry is required to perform the monitoring: 1) it lowers the overhead of QoS registry and simplifies the implementation of QoS registry, 2) data is collected from actual consumption of the service which is up to date and objective, 3) it avoids the necessity to install expensive middleware to poll the large number of service providers constantly.

  Through active execution monitoring, when a service requester gets a different set of values for those deterministic criteria advertised by the service provider, this difference can be logged. If this phenomenon is common in a particular domain, we can expand the criteria for `Reputation` to record the difference between the actual deterministic quality criteria and the advertised deterministic quality criteria from the service provider. The bigger the difference, the lower the QoS will be for that service provider.

- **Collecting quality information from users feedback.** Each end-user is required to update QoS of service he/she has just consumed. This ensures that the QoS registry is fair to all end-users since QoS values can be computed based on real user experience with up to date runtime execution data. To prevent the manipulation of QoS by a single party, for each feedback, the end-user is given a pair of keys. This pair of keys must be authenticated by the service provider before the user is allowed to update the QoS value. The update must take place in a limited time-frame to prevent the system from being overloaded with un-consumed service requests.

We assume that service requester can either download a plug-in from QoS registry for providing feedback on QoS or access the services via a portal which must implement the active monitoring mechanism as outlined above for QoS computation.

4. EXPERIMENTS

We conducted a series of experiments to: 1) investigate the relationship between QoS value and the business criteria, 2) study the effectiveness of price and the service sensitivity factors in our QoS computation. The experiments are conducted on a Pentium computer with a 750Mhz CPU and 640MB RAM. We first simulated 600 users searching for services, consuming the services and providing feedbacks to update the QoS registry in the UPS application. This will

Providers	**Price**	**Transaction**	**Time out**	**Compensation Rate**	**penalty Rate**	**Execution Duration**	**Reputation**
ABC	25	yes	60	0.5	0.5	100	2.0
BTT	40	yes	200	0.8	0.1	40	2.5

Table 1: Comparing Service Quality Data of two providers

generate a set of test data for those non-deterministic quality criteria such as reputation and execution duration for each service provider in the QoS registry. The deterministic quality criteria (price, penalty rate) have been advertised and stored in the QoS registry for each service provider prior to the simulation.

Table 1 shows the values of various quality criteria from two phone service providers with respect to local phone service. From this table, we can see that provider ABC has a better price but the various criteria that are related to services are not very good. Its time out value is small, this means it is not so flexible for end-users. Its compensation rate is lower than BTT, and its penalty rate is higher than BTT. Its execution time is longer than BTT and reputation is lower than BTT as well. Using our QoS computation algorithm, can we ensure that ABC will win for a price sensitive search or BTT will win for a service sensitive search?

Table 2 shows that BTT has a QoS value of 3.938 with a price sensitivity search and a QoS value of 5.437 with a service sensitivity search. On the other hand, ABC has a QoS value of 4.281 with a price sensitivity search and a QoS value of 3.938 with a service sensitivity search. ABC is winning for a price sensitive search ($4.281 > 3.938$), BTT is winning for a service sensitive search ($5.437 > 4.281$) which verifies our original hypothesis.

The following three figures show the relationship between the QoS value and the price, the compensation rate, the penalty rate, and the sensitivities factors. From Figure 2, we can see that the QoS increases exponentially as the price approaches zero. As the price reaches 9x20 = 180, the QoS becomes very stable. This gives us the confidence that price can be used very effectively to increase QoS within certain range. It also shows that in our QoS computation model, QoS cannot be dominated by price component indefinitely.

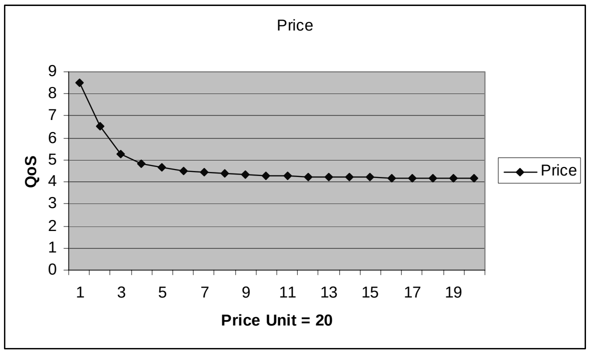

Figure 2: *Relationship between QoS and Price.*

Figure 3 shows the relationship between QoS and services. For the penalty, it shows that the QoS value decreases as the penalty rate increases. For the compensation, it shows that the QoS value increases as the compensation rate increases. This concludes that QoS computation cannot be dominated by these two components indefinitely.

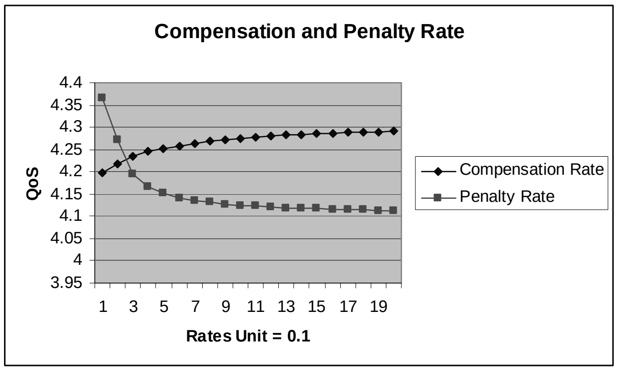

Figure 3: *Relationship between QoS, Compensation and Penalty Rate.*

Figure 4 indicates that QoS value increases almost four times faster for the service sensitivity than the price sensitivity. This shows that using the the same sensitivity value for both price and service factors will not be effective for the price sensitivity search in our QoS registry. To get an effective price and service sensitivity factors for a domain, we need to analyze the sensitivity factors after the QoS registry has been used for a while and re-adjust the values by performing experiment mentioned in Figure 4.

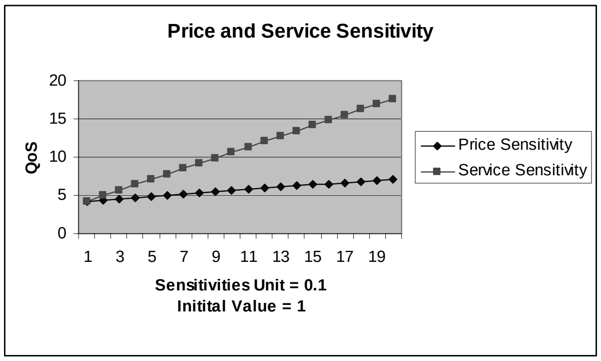

Figure 4: *Relationship between QoS, Price and Service Sensitivity factors.*

Providers	Price Sensitivity	Service Sensitivity	QoS Value
ABC	1	2	4.675
BTT	1	2	5.437
ABC	2	1	4.281
BTT	2	1	3.938

Table 2: Results of Sensitivity Experiments

5. RELATED WORK

Multiple web services may provide similar functionality, but with different non-functional properties. In the selection of a web service, it is important to consider both functional and non-functional properties in order to satisfy the constraints or needs of users. While it is common to have a QoS model for a particular domain, an extensible QoS model that allows addition of new quality criteria without affecting the formula used for the overall computation of QoS values has not been proposed before. Moreover, very little research has been done on how to ensure that service quality criteria can be collected in a fair, dynamic and open manner. Most previous work is centered on the collection of networking level criteria such as execution time, reliability and availability etc [12]. No model gives details on how business criteria such as compensation and penalty rates can be included in QoS computation and enforced in an objective manner.

In a dynamic environment, service providers can appear and disappear around the clock. Service providers can change their services at any time in order to remain competitive. Previous work has not addressed the dynamic aspects of QoS computation. For example, there is no guarantee that the QoS obtained at run time for a particular provider is indeed the most up to date value. Our proposed QoS computation which is based on active monitoring and consumers' feedback ensures that QoS value of a particular provider is always up to date.

In [9], the author proposed a QoS model which has a QoS certifier to verify published QoS criteria. It requires all web service providers to advertise their services with the QoS certifier. This approach lacks the ability to meet the dynamics of a market place where the need of both consumers and providers are constantly changing. For example, it does not provide methods for providers to update their QoS dynamically, and does not give the most updated QoS to service consumers either. There are no details on how to verify the QoS with the service providers in an automatic and cost effective way.

In [10], the authors proposed a QoS middleware infrastructure which required a build-in tool to monitor metrics of QoS automatically. If such a tool can be built, it needs to poll all web services to collect metrics of their QoS. Such an approach requires the willingness of service providers to surrender some of their autonomy. Moreover, if the polling interval is set too long, the QoS will not be up to date. If the polling interval is set too short, it might incur a high performance overhead. A similar approach which emphasizes on service reputation is proposed in [5, 4]. A web service agent proxy is set up to collect reputation rating from previous usage of web services. The major problem with this approach is that, there is no mechanism in place to prevent false ratings being collected. We have a mechanism to ensure that only the true user of the service is allowed to provide feedback.

In [3], the author discusses a model with service level agreement (SLA) which is used as a bridge between service providers and consumers. The penalty concept is given in its definition of SLA. The main emphasis about this concept is what should be done if the service provider can not deliver the service under the defined SLA, and the options for consumers to terminate the service under the SLA. Penalty is not included as a service quality criteria which can be used in QoS computation. In [2], a language-based QoS model is proposed. Here, QoS of a web service is defined using a Web Service QoS Extension Language. This XML-based language can define components of QoS in a schema file which can be used across different platforms and is human readable.

6. CONCLUSION

The quality-based selection of web services is an active research topic in the dynamic composition of services. The main drawback of current work in dynamic web service selection is the inability to ensure that the QoS of published web services remain open, trustworthy and fair. We have presented an extensible QoS model that is open, fair and dynamic for both service requesters and service providers. We achieved the dynamic and fair computation of QoS values of web services through a secure active users' feedback and active monitoring. Our QoS model concentrates on criteria that can be collected and enforced objectively. In particular, we showed how business related criteria (compensation, penalty policies and transaction) can be measured and included in QoS computation. Our QoS model is extensible and thus new domain specific criteria can be added without changing the underlying computation model. We provided an implementation of a QoS registry based on our extensible QoS model in the context of a hypothetical phone service provisioning market place. We also conducted experiments which validate the formula we used in the computation of QoS values in the phone service provisioning domain.

The formula for computing the QoS values can be varied for different domains by using different sensitivity factors or different criteria with its associated groupings. Additional factors, different groupings, or criteria might serve certain domains better. We provided a mechanism for service providers to query their QoS computed by the registry, and update their published services to become more competitive at anytime.

The success of this model depends on the willingness of end-users to give feedback on the quality of the services that they consume. Our future work includes ways to automate the collection of feedback data. To ensure that all users will provide feedback, we need to make providing feedback very straight forward. We also want to see that the implemented QoS registry becomes intelligent internally and externally. Internal intelligence refers to the ability to infer the appropriate sensitivity factors to use to get the list of providers which meets end-users requirements. External intelligence refers to the ability of QoS registry to be able to predict the trend of QoS from certain providers at runtime. For example, we want to have a system which knows how much the QoS value decreases in the morning and increases in the evening or during different time of the year, and takes action to notify its users. For the service provider, we want to have the ability to notify the provider when the QoS of its service is below a certain threshold.

7. REFERENCES

[1] Boualem Benatallah and Fabio Casati, editors. *Distributed and Parallel Database, Special issue on Web Services*. Springer-Verlag, 2002.

[2] Peter Farkas and Hassan Charaf. Web services planning concepts. *Journal of WSCG*, 11(1), February 2003.

[3] Li jie Jin and Vijay Machirajuand Akhi Sahai. Analysis on Service Level Agreement of Web Services. Technical Report HPL-2002-180, Software Technology Laboratories, HP Laboratories, June 2002.

[4] E. Michael Maximilien and Munindar P. Singh. Conceptual Model of Web Services Reputation. *SIGMOD Record*, October 2002.

[5] E. Michael Maximilien and Munindar P. Singh. Reputation and Endorsement for Web Services. *ACM SIGecom Exchanges*, 3(1):24–31, 2002.

[6] Daniel A. Menasce. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6), 2002.

[7] J. O'Sullivan, D. Edmond, and A. ter Hofstede. What's in a Service? *Distributed and Parallel Databases*, 12(2–3):117–133, September 2002.

[8] M.P. Papazoglou and D. Georgakopoulos. Serive-Oriented Computing. *Communcications of the ACM*, 46(10):25–65, 2003.

[9] Shuping Ran. A Model for Web Sevices Discovery with QoS. *ACM SIGecom Exchanges*, 4(1):1–10, 2003.

[10] Amit Sheth, Jorge Cardoso, John Miller, and Krys Kochut. Qos for service-oriented middleware. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics abd Informatics (SCI02)*, pages 528–534, July 2002.

[11] Aad van Moorsel. Metrics for the Internet Age: Quality of Experience and Quality of Business. Technical Report HPL-2001-179, HP Labs, August 2001. Also published in 5th Performability Workshop, September 2001, Erlangen, Germany.

[12] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality Driven Web Services Composition. In *Proceedings of the 12th international conference on World Wide Web (WWW), Budapest, Hungary.* ACM Press, May 2003.