International Journal of Computer Applications (0975 – 8887)
Volume 26– No.1, July 2011

A Publish/Subscribe Model for QoS-aware Service Provisioning and Selection

Elarbi Badidi
Faculty of Information Technology,
United Arab Emirates University, P.O. Box 17551,
Al-Ain, United Arab Emirates

ABSTRACT

With the growing adoption of the Service Oriented Architecture (SOA) in the industry and the wide deployment of Web services, users are increasingly requiring services that are capable of meeting their quality-of-service (QoS) requirements. In this paper, we propose a novel framework for QoS-aware Web service provisioning, which relies on QoS brokers, to mediate between clients and service providers, and a QoS Notification Broker that implements a publish/subscribe model to handle notifications on significant changes in QoS offerings. Furthermore, we describe a multi-attributes algorithm for the selection of potential service providers that can fulfill clients' requests. The algorithm calculates the utility value of each service provider, per Web service type, based on the client QoS requirements. One of the advantages of the approach is that service providers may provide several service types. These services may be simple Web services or composite Web services aggregated from other services. The publish/subscribe model allows QoS brokers to be aware of significant changes in the QoS offerings of service providers; and consequently, be able to make informed selection decisions. Besides, the proposed selection algorithm allows ranking service providers by matching their up-to-date QoS offers against the QoS required by the client.

Keywords

Web services; Service Oriented Architecture; QoS; QoS management; QoS Broker; Notification broker.

1. INTRODUCTION

The proliferation of broadband, wireless, and cellular networks has led to a remarkable rise in the number of users who are using a variety of modern Internet-enabled devices to consume online business services. Service oriented computing and Web technologies facilitate the deployment of business applications on the web, collaboration among businesses, and application integration on a global scale. The current most promising technology to rely on the idea of service oriented computing is Web services technology. It provides the basis for the development, the deployment, and the invocation of business processes, distributed over the Internet, via standard APIs (Application Programming Interfaces) and protocols.

As a result of this rapid growth, users increasingly require services that can meet their QoS requirements. Thus, businesses should provide QoS-aware services if they want to remain competitive. Most research work on the support of QoS in SOA focused on identifying QoS requirements and mechanisms for QoS management. Numerous efforts have investigated approaches for describing QoS offerings of Web services and for publishing QoS-aware Web services. This includes the Web Services Management Framework (WSMF) [1] and the Web Services Offer Language (WSOL) [2]. Furthermore, many frameworks and middleware infrastructures have been proposed to provide support for the management ofthe QoS of Web services and to provide users with QoS-aware services [3][4][5]. The authors in [3] have designed and implemented a QoS brokerage system, which monitors QoS with respect to availability, performance, and reliability of Web services. In [4], the authors proposed a framework that is relying on a QoS broker for the composition of QoS-aware Web services. The broker's components implement dynamic service composition, service selection, and service adaptation. The authors in [5] described a Web service framework that supports QoS management using QoS brokers. Clients interact with the UDDI (Universal Description, Discovery and Integration) registry through the QoS brokers. The brokers publish QoS information they obtain from service providers in the UDDI and help clients choose services according to their functional and QoS needs.

A key aspect of QoS management, which is not addressed adequately by most QoS management systems, is the management of the continual change in the QoS delivered by service providers. This change is mainly due to the variation in workload. Clients are notified of these changes only after a certain period and can suffer degradation in the QoS they are expecting from service providers.

To cope with the issues of QoS-aware service provisioning, QoS-driven selection of service providers, and management of the continual change in QoS offerings, we propose a novel framework for QoS management. The main components of the framework are QoS brokers and a QoS Notification Broker. QoS brokers mediate between clients and service providers with regards to the services and the QoS that service providers should deliver. They receive notifications on any significant change in the QoS offering of service providers by means of the QoS Notification Broker, which implements a publish/subscribe model. Furthermore, we describe a multi-attributes decision algorithm for the selection of service providers by QoS brokers on the basis of the QoS they can offer and the client QoS requirements.

The remainder of the paper is organized as follows. Section 2 provides background information on the concept of QoS and the various models for QoS information representation. Section 3 presents an overview of our proposed framework, and describes the programming interfaces of the framework's components then the interactions among them. Section 4 describes our proposed multi-attributes decision algorithm for the selection of QoS-aware service providers. Section 5 discusses some issues and challenges of the approach. Finally, Section 6 concludes the paper and describes future work.

2. BACKGROUNDAND RELATED WORK

2.1 Quality-of-Service in SOA

The term "QoS" originates from the fields of telecommunications, distributed multimedia, and networking. QoS refers to a collection of qualities or characteristics of a service, such as availability, security, response-time, throughput, latency, reliability, and reputation. The arrangement between the customer and the service provider is referred to as the Service Level Agreement (SLA). An SLA describes agreed service functionality, cost, and qualities [6]. Availability represents the percentage of time that the Web service is operating. Security characteristics comprise the authentication mechanisms that the service offers, encryption, and access control. The Web service provider may offer different security levels depending on the client's request. Response-time is the time a service takes to respond to diverse types of requests. Throughput is the speed at which a service can process requests. Latency is the elapsed time between sending a request and receiving the response. Reputation is a qualitative measure of web services trustworthiness. It depends on the end-users' experiences in using a Web service [7].

2.2 QoS Representation Models

Various models and approaches have been proposed in literature for representing QoS parameters in SOA and for providing QoS support in Web services. The most significant ones are:

- Extension of WSDL (Web Services Description Language) with QoS information
- Extension of UDDI with QoS information
- Utilization of a QoS broker
- Utilization of WS_Policy

2.2.1 WSDL Extension

The initial specification of WSDL does not provide support for the description of nonfunctional properties of a Web service. Several proposals for extending WSDL with QoS information have been proposed [8][9]. Kang [9] advocates the use of annotations, which WSDL supports, to describe QoS information in a WSDL document. In [8], the authors proposed extension of WSDL with QoS information using a meta-model transformation, which is consistent with the Model Driven Architecture (MDA) principles and recommendations. The WSDL meta-model is then transformed into a QoS-enabled WSDL (Q-WSDL) meta-model, which can be used to specify QoS attributes.

Another relevant research work regarding the integration of QoS attributes in WSDL is the work of WSQM TC, an OASIS technical committee for Web service quality model, which published the Web Services Quality Model (WSQM). WSQM models and explains the quality factor, quality action, and quality attributes for Web services. The authors in [10] describe the WSQDL (Web Services Quality Description Language), which uses WSQM.

2.2.2 UDDI Extension

This approach consists to extend the current UDDI data structure with QoS information of a Web service. Many research works advocate the utilization of tModels structures to express QoS attributes [11][12][13]. In [11], the authors proposed three approaches, namely type-based, keyword-based and ontological-based approaches, to model QoS tModel (Technical Model) that can be stored in the UDDI. In [13], the KeyName attribute of the tModel holds the name of a quality attribute while the KeyValue attribute holds its values. Blum et al. [12] submitted their work to OASIS UDDI Specification TC for standardization. A. Shaikh et al. [14] developed a compliant UDDI, called UDDIe that allows storing QoS attributes within the property bag element.

2.2.3 Utilization of a Brokerage Service

This approach has been used in several works [15][16][3][4][5][17]. The QoS broker acts as an intermediary between clients and service providers. The functions of the QoS broker typically include monitoring and collecting QoS information of Web services, making selection decisions on behalf of clients, and negotiating SLAs and QoS assurances with Web services.

2.2.4 Utilization of WS_Policy

Other works [18] [19] have proposed extensions to the Web services Policy Framework (WS-Policy) to represent QoS policies of Web services [20]. WS_Policy does not define how policies are discovered or attached to a Web service. The WS_PolicyAttachment specification [21] defines such mechanisms, especially for associating policy with WSDL artifacts and UDDI elements.

The above approaches can be combined to enable better support of QoS in SOA. Furthermore, some efforts have added further structure, specifically QoS constraints, to Web services descriptions through the use of ontologies [22] [23].

3. A FRAMEWORK FOR QOS-AWARE SERVICE PROVISIONING

In any business activity with a system of delivery and consumption, brokers emerge to facilitate business between consumers and providers. This is the case for service delivery in a SOA and Web-based environment. QoS brokers can be used to decouple clients from service providers while managing the provisioning of QoS.

Figure 1 depicts our framework for Web service provisioning. The main components of the framework are clients, QoS brokers, QoS Notification Broker, and Service providers. Multiple QoS brokers may be deployed, one for each local domain for instance. A discovery service will allow clients to bind to the right QoS broker.

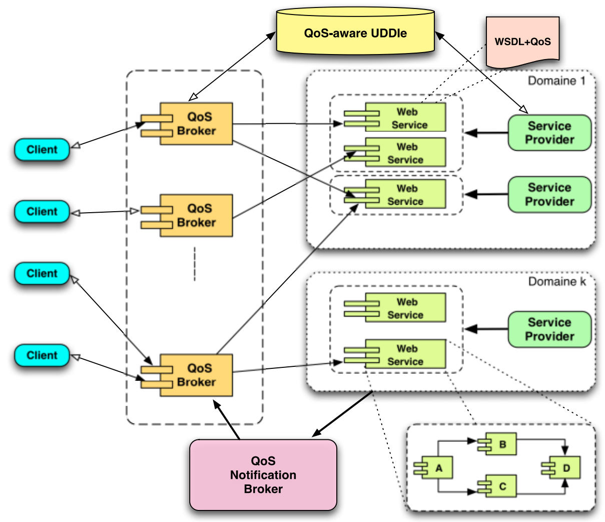

Figure 1: Framework for QoS-aware service provisioning

3.1 QoS Brokers

A QoS broker is a mediator service that decouples clients from service providers. It is in charge of handling subscriptions of clients in which they express their interest to consume some type of service, and registration of service providers that are willing to provide some types of service. The QoS broker may also find service providers that offer a certain service type by looking up an UDDIe directory [14]. In addition to the basic functionalities of a traditional UDDI server, the UDDIe server provides support for the specification of the QoS that a service provider can ensure to its clients. Given that Web services providers and clients do not normally have the capabilities to negotiate, manage, and monitor QoS, they delegate management tasks, such as Web services selection and QoS negotiation, to the QoS Broker. QoS brokers are aware of the current QoS of service providers through a QoS Notification Broker that implements a topic-based publish-subscribe system.

Figure 2 shows the architecture of the QoS broker, which includes several components that cooperate in order to deliver personalized services to clients with various devices. These components are the Request Dispatcher, the QoS Negotiator, the QoS Information Manager, the Profile Manager, and the Policy Manager. They are under the control of the Coordinator component. They allow carrying out various management operations such as admission control, QoS-based service selection,QoS negotiation, user profile management, and policies management. The back-end databases maintain information about services' policies, clients' profiles and preferences, and dynamic QoS information.

The *Request Dispatcher* is in charge of the admission control of incoming requests by determining whether the received requests can use the requested services. The Request Dispatcher is also in charge of implementing different policies for the selection of service providers, based on the client's QoS requirements and the service providers' QoS offerings. We describe a new algorithm for QoS-aware server selection in Section 4. The algorithm takes account of the current conditions and capabilities of potential service providers as well as the QoS required by the client and his/her weights for quality attributes.

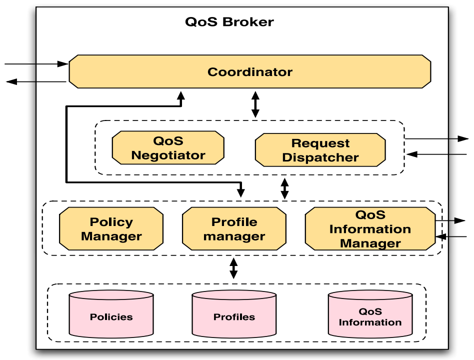

Figure 2: Architecture of the QoS Broker

The *QoS Negotiator* is in charge of carrying out the negotiation process in order to reach an agreement as to the QoS to be delivered to the client. First, the client notifies the QoS broker about its required service and its preferred level of QoS. Based upon available QoS information, the Request Dispatcher selects an appropriate service provider, according to the selection policy, capable of satisfying the required QoS. Then, the *QoSNegotiator* approaches this service provider to determine whether it can ensure the required level of QoS given its current conditions. Afterwards, the client and the service provider sign a contract. The contract specifies the service type that the provider should offer to the client, the QoS to ensure, the cost of service, and actions to take when there is a violation of the agreement on QoS. If the selected service provider is unable to deliver the required QoS, the broker selects another service provider and reiterates the negotiation process.

The *Profile Manager* is responsible for managing clients' profiles, including their preferences in terms of personalized services and required QoS.

The *Policy Manager* is responsible for managing different kinds of policies such as authorization policies and QoS-aware selection policies of service providers.

3.2 QoS Notification Broker

The QoS Notification Broker implements a topic-based publish/subscribe system in which service providers are the publishers and QoS brokers are the subscribers. Figure 3 depicts this model. QoS offerings of Web service types, requested by clients, represent the topics of the system. The Publish/subscribe messaging model is a one-to-many pattern of asynchronous message distribution based on registration of interest. In this model, publishers associate the name of a topic to each message ("publish") rather than addressing it directly to subscribers. Then, the message system sends the message to all eligible recipients that expressed their interest in receiving messages on that topic ("subscribe"). As opposed to point-to-point messaging systems, such as message queuing, the publish/subscribe model of asynchronous communication is a far more scalable architecture. This is because the source of the information has only to concern itself with creating the information, and can leave the task of servicing potential recipients to the messaging system. It is a loosely coupled architecture in which senders often do not need to know who their potential subscribers are, and the subscribers do not need to know who generates the information.

Web Service Type	WST quality offering	Service Provider	QoS Broker
WST1	a	SP1, SP2	QoSB1, QoSB3
WST2	b	SP1, SP3	QoSB1, QoSB2, QoSB3
WST3	c	SP2	QoSB2, QoSB3

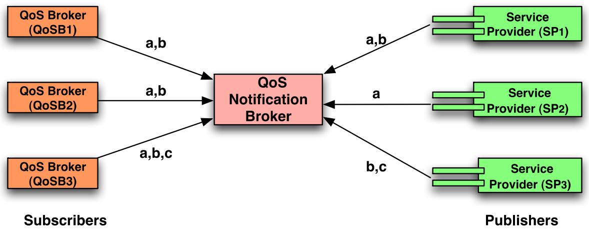

Figure 3: Topic-based publish/subscribe system

In addition to this model for getting QoS updates, the QoS Notification Broker implements a normal on-demand request/response model, in which it requests up-to-date QoS offering from service providers once a QoS broker requires QoS information for a given Web service type. Therefore, the QoS Notification Broker may either pull QoS offering from service providers or let service providers push updated QoS offering.

Service providers, typically residing in different domains, deliver services to clients with various QoS. Therefore, a QoS broker is in charge of selecting appropriate QoS-aware service providers to deliver services requested by a client. In section 4, we describe our proposed selection algorithm that allows the ranking of service providers based on the QoS they can offer and the QoS required by the client.

3.3 Service Providers

As shown in Figure 1, service providers can offer several types of services using Web services. These Web services can be simple or composite Web services that are the result of the composition of many simple or composite services. In order to estimate their current QoS for each service type they offer, service providers should use monitoring techniques that allow collecting measurement data at selected observation points. By aggregating collected data, the service provider can determine the value of each QoS indicator. If there is a significant change in the current QoS of services, the service provider notifies the QoS Notification Broker about the change in its QoS offering. Then, the QoS Notification Broker notifies any subscriber to the corresponding QoS offering of that change.

Figure 4 depicts the process of QoS evaluation and notification at the service provider site. Monitoring data is collected at various points of observation. It is then used by the QoS Evaluator component in order to provide an estimation of the current QoS offerings. QoS Information is made available to the QoS Negotiator and QoS Notifier components. The QoS Negotiator is responsible for negotiating with QoS brokers, or directly with clients, the service and the QoS level to be delivered. The QoS Notifier is in charge of notifying the QoS Notification Broker of substantial changes in the QoS offering of a given service type.

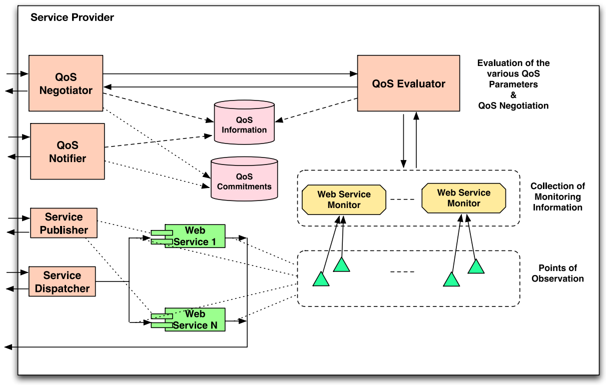

Figure 4:Service provider architecture

3.4 Interfaces and Interaction Model

To describe the interactions between the components of the framework, we consider only the case of a single QoS broker. The model can be easily extended to consider several QoS brokers. Figure 5 depicts the interfaces of the framework's components, and Figure 6 shows the interactions among them.

The QoS Notification Broker acts as an intermediary between publishers (service providers) and subscribers (QoS brokers) on a collection of Web service types (QoS offerings).

A QoS broker invokes the *registerSubsriber()* method of the QoS Notification Broker to register its interest in using the services of the QoS Notification Broker. If the processing of this method is successful, the QoS Notification Broker returns a subscription ID to the QoS broker that will be used as parameter in subsequent requests for service.

The QoS broker invokes the *subscribe()* method of the QoS Notification Broker to register its interest to receive updates on QoS offerings of some service types. Conversely, the QoS broker may invoke the *unsubscribe()* method of the QoS Notification Broker if it is not interested anymore in receiving updates on the QoS offering of a Web service type.

Similarly, a service provider invokes *registerPublisher()* of the QoS Notification Broker to register its interest to publish QoS offering of some types of Web service through the QoS Notification broker. If the processing of that method is successful, the QoS Notification Broker returns a registration ID to the service provider that is used in subsequent requests of the service provider.

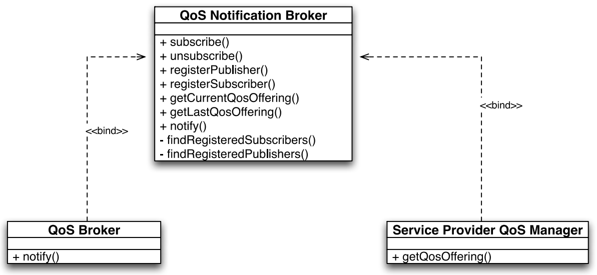

Figure 5:Interfaces of the framework's components.

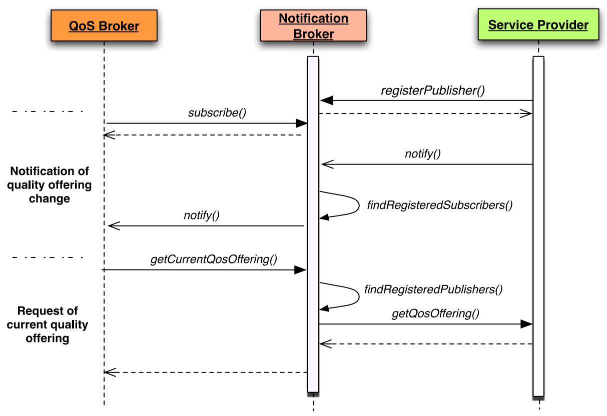

Figure 6: Diagram of interactions among the framework components

The QoS Notification Broker receives notifications on QoS offering change through its *notify()* method that a service provider invokes. It, then, notifies a QoS broker about that change by invoking its *notify()* method. Furthermore, a QoS broker may request the current value for a given Web service type by invoking *getCurrentQosOffering()* of the QoS Notification broker, which forwards the request to service providers that are providing that Web service type requested by the QoS broker. A newly-subscribed QoS broker can invoke *getLastQosOffering()* in order to get the last value of a given Web service type that other QoS brokers have already received.

The QoS Notification Broker has also two private additional methods *findRegisteredSubscribers()* and *findRegisteredPublishers()*. The first method is invoked to get the list of QoS brokers, which have subscribed to a given Web service type. The Notification broker calls this method once it has received a notification of QoS offering change for that Web service type. The second method is invoked to get the list of service providers that are publishing the Web service type requested by a QoS broker that has invoked *getCurrentQosOffering()*.

This mode of interaction allows service providers to notify QoS brokers about any substantial change in their QoS offerings. In the same way, it allows QoS brokers to request information regarding the QoS offering of service providers that are offering a service type requested by the client. Using up-to-date information on QoS offerings allows QoS brokers to make informed decisions during the selection of appropriate service providers, which can fulfill the QoS requirements of a client.

3.5 A MULTI-ATTRIBUTES SELECTION ALGORITHM

As we have stated earlier, the QoS broker is in charge of selecting appropriate service providers to deliver services requested by the client. Several service providers may provide the same service to the client. Thus, the selection has to be done according to the service providers' QoS offerings and the client's QoS requirements. In this section, we describe our proposed algorithm for the selection of service providers, which relies on using multi-attributes utility functions. We describe how the algorithm works in the case of a single domain. However, it can be easily extended to the case of multiple domains as depicted by Fig. 1.

As numerous potential service providers, within the domain, can deliver the Web service required by a consumer, it is essential to consider only potential service providers that can satisfy the QoS required by the client.

Let $Q = \{Q_1, Q_2, \dots, Q_P\}$ be the list of QoS indicators considered in the system. QoS indicators may concern for instance parameters such as *availability*, *throughput*, *response time*, *reputation*, and *cost of service*. Let $T = \{t_1, t_2, \dots, t_N\}$ be the list of Web service types to which a client has subscribed by showing its interest in receiving their QoS offerings. The client may, for example, require these service types to create a composite service.

Let $SP = \{SP_1, SP_2, \dots, SP_K\}$ be the list of service providers, which have subscribed with the *QoS Notification broker*. Two service providers may provide similar or different Web service types. One service provider offers, for example, *flight booking* and *hotel reservation* services while the other provider offers only hotel reservation.

For a given service type, candidate service providers typically provide service functionality with different QoS. The following vector expresses the QoS offer of a service provider SP_r for a Web service type t_j.

$$Q_j^r = \{Q_{1,j}, Q_{2,j}, \dots, Q_{P,j}\}$$

To enable sorting and ranking of service provider candidates, we consider normalized values of the QoS offers and utility functions to map the vector of QoS values into a single real value. We define

$$q_{i,j}^r = \frac{Q_{i,j}^{max} - Q_{i,j}^r}{Q_{i,j}^{max} - Q_{i,j}^{min}}$$

as the normalized value of the ith quality indicator for service type t_j by the service provider SP_r .

$$Q_{i,j}^{max} = \max_{\substack{1 \le i \le P \\ 1 \le j \le N}} (Q_{i,j})$$

$$Q_{i,j}^{min} = \min_{\substack{1 \le i \le P \\ 1 \le j \le N}} (Q_{i,j})$$

For each QoS offering, to which the client subscribed, the client specifies the *min* values of the normalized QoS indicators that he can tolerate. The following vector expresses the minimum QoS requirements that the client tolerates for a given Web service type $t_j$with $1 \le j \le N$:

$$M_j = \{m_{1,j}, m_{2,j}, \dots, m_{P,j}\}$$

$m_{i,j}$is the minimal acceptable value of $Q_i$for service type t_j. $0 \le m_{i,j} \le 1$, $1 \le i \le P$and $1 \le j \le N$.

Therefore, the following matrix expresses the whole QoS requirements of the client for all its subscribed Web service types and all QoS indicators considered in the system:

$$M = \begin{bmatrix} m_{1,1} & m_{2,1} \cdots & \cdots & m_{P,1} \\ \vdots & & & \vdots \\ m_{1,2} & m_{2,2} \cdots & \cdots & m_{P,2} \\ \vdots & & & \\ m_{1,N} & m_{2,N} \cdots & \cdots & m_{P,N} \end{bmatrix} \qquad (1)$$

A zero value in the matrix means that the client has no constraint on the corresponding QoS parameter. The algorithm aims to find for each Web service type t_j, to which the client subscribed, a suitable service provider, which can meet the minimum quality requirements of the client. The following matrix expresses the QoS offer of a service provider SP_r.

$$T_r = \begin{bmatrix} q^r_{1,1} & q^r_{2,1} \cdots & \cdots & q^r_{P,1} \\ \vdots & & & \vdots \\ q^r_{1,2} & q^r_{2,2} \cdots & \cdots & q^r_{P,2} \\ \vdots & & & \\ q^r_{1,N} & q^r_{2,N} \cdots & \cdots & q^r_{P,N} \end{bmatrix} \qquad (2)$$

$q^r_{i,j}$is the value of $Q_i$for service type t_j that SP_rcan assure. $0 \le q^r_{i,j} \le 1$, with$1 \le i \le P$ and $1 \le j \le N$

$SP_r$is suitable for provisioning Web service type t_j if the client's minimum QoS requirements are satisfied. This means that:

$$0 \le m_{i,j} \le q^r_{i,j} \le 1 \text{for} 1 \le i \le P \text{ and } 1 \le j \le N$$

The client can assign relative weights to the QoS indicators. He may even set weights for each Web service type to which it subscribed. For example, for the *flight booking* Web service type, more weight may be given to the *availability* indicator than to the *cost* indicator. For the *hotel reservation Web service type*, more weight may be given, for example, to the *cost* indicator than to the other QoS indicators. Therefore, the weight matrix is given by:

$$W = \begin{bmatrix} w_{1,1} & w_{2,1} \cdots & \cdots & w_{P,1} \\ \vdots & & & \vdots \\ w_{1,2} & w_{2,2} \cdots & \cdots & w_{P,2} \\ \vdots & & & \\ w_{1,N} & w_{2,N} \cdots & \cdots & w_{P,N} \end{bmatrix} \qquad (3)$$

$w_{i,j}$is the weight given to quality indicator Q_i for service type t_j. $1 \le i \le P$and$1 \le j \le N$ and and $\sum_{i=1}^{P} w_{i,j} = 1$

The score of a given QoS indicator $Q_i$for a given Web service type $t_j$by SP_r offer is:

$$s^r_{i,j} = w_{i,j} \times q^r_{i,j} \text{for} 1 \le i \le P \text{ and } 1 \le j \le N$$

The score matrix S_r of SP_roffer, for all QoS indicators and all Web service types of the system is:

$$S_r = \begin{bmatrix} s^r_{1,1} & s^r_{2,1} \cdots & \cdots & s^r_{P,1} \\ \vdots & & & \vdots \\ s^r_{2,1} & s^r_{2,2} \cdots & \cdots & s^r_{P,2} \\ \vdots & & & \\ s^r_{1,N} & s^r_{2,N} \cdots & \cdots & s^r_{P,N} \end{bmatrix} \qquad (4)$$

Given the weight matrix and the minimum QoS requirements matrix, the minimum score matrix is:

$$S_{min} = \begin{bmatrix} l_{1,1} & l_{2,1} \cdots & \cdots & l_{P,1} \\ \vdots & & & \vdots \\ l_{1,2} & l_{2,2} \cdots & \cdots & l_{P,2} \\ \vdots & & & \\ l_{1,N} & l_{2,N} \cdots & \cdots & l_{P,N} \end{bmatrix} \qquad (5)$$

Where $l_{i,j} = w_{i,j} \times m_{i,j}$

for$1 \le i \le P$ and $1 \le j \le N$

The difference matrix, $S_r - S_{min}$, shows whether SP_rcan satisfy or not all QoS requirements for all Web service types to which the client has subscribed. A value that is less than zero in this matrix means that SP_rcannot satisfy the QoS requirements for the corresponding Web service type and QoS indicator. Therefore, we have to reason per Web service type, and consider only service providers that can meet the QoS requirements for that Web service type. The utility function per Web service type $t_j$for a candidate service provider SP_r offer is:

$$U^r_j = \sum_{i=1}^{P} s^r_{i,j}. \qquad (6)$$

This value corresponds to the linear additive utility function. The following vector expresses the utility vector of SP_r for all Web service types:

$$U_r = \begin{bmatrix} U^r_1 \\ U^r_2 \\ \cdots \\ \cdots \\ U^r_N \end{bmatrix} \qquad (7)$$

Considering the utility functions of the entire candidate service providers, we get the following decision matrix:

	SP_1	SP_r	…	SP_K	Max Utility	Selected SP
t_1	U^1_1	U^r_1	…	U^K_1	…	…
t_j	U^1_j	U^r_j	…	U^K_j	…	…
…	…	…	…	…	…	…
t_N	U^1_N	U^r_N	…	U^K_N	…	…

A zero in the decision matrix means that the corresponding service provider cannot meet the QoS requirements of the corresponding Web service type. The maximum value of all utility functions in a row *j* corresponds to the best QoS offer that can fulfill the QoS requirements of the client for the Web service type t_j. The *most appropriate service provider* (MASP) for delivering Web service type t_j is the provider that maximizes the above utility functions.

$$MASP_j \leftarrow \max_{1 \le r \le K}(U^r_j). \qquad (8)$$

If no service provider meets the client's QoS requirements for a given Web service type, then the *QoS broker* may ask the client to lower its QoS expectations.

Figure7 summarizes the steps of the algorithm.

Step-1: Construct the normalized matrix $\mathbf{M}$, defined in (1), of the client's minimum QoS requirements.

Step-2: Construct the client's weight matrix **W**, defined in (3), and the minimum score matrix $\mathbf{S_{min}}$, defined in (5).

Step-3: For each candidate Service provider $\mathbf{SP_r}$ registered with the QoS broker,

a) Construct the normalized matrix $\mathbf{T_r}$, defined in (2), of the QoS offering of SP_1.
b) Calculate the score matrix $\mathbf{S_r}$, defined in (4), that represents the score of the QoS offering of SP_1 against the client QoS requirements for each Web service type.
c) Calculate $\mathbf{S_r} - \mathbf{S_{min}}$. If a value of this matrix is less than zero, then it means that SP_1 cannot satisfy the QoS requirements of the client for the associated Web service type and the associated QoS indicator. Only rows with positive values will be considered in the next steps.
d) Calculate the Utility vector $\mathbf{U_r}$, defined in (7). Note that rows with negative values in the difference matrix will have a score 0 in the decision matrix created in step 4.

Step-4: Create the decision matrix, and fill out the maximum utility value for each Web service type and the SP providing that value.

The most suitable service provider (MSSP) for each Web service type is given by equation (8).

Figure 7: QoS-based service providerselection algorithm

4. ISSUES AND CHALLENGES

Performance monitoring, billing, managing clients' expectations are significant concerns among others that a service provider has to handle. The service provider must ensure that its services are highly available and that its clients can access them. Security is also a prime concern with any application service. Therefore, the service provider should design and implement simple and efficient security solutions such as an identity management service. In this scenario, each client of the service provider has an identity account, which the system uses to authenticate the client and track all his requests for service.

In the proposed model, the QoS Notification Broker sends notifications to QoS brokers by relying on the good will of service providers for signaling any significant change in their QoS. Both QoS brokers and the QoS Notification Broker do not have the necessary tools to carry out independent monitoring of the QoS delivered by service providers. Therefore, it is necessary to extend the framework by implementing some form of QoS monitoring at the QoS broker level in order to evaluate the real QoS delivered by a service provider. Moreover, we have deliberately considered only a single QoS Notification Broker in the framework in order to show how QoS brokers become aware of changes in the QoS offered by service providers. The model can be easily extended to support multiple QoS Notification brokers.

Another concern that should be handled by QoS brokers and the QoS Notification Broker is the heterogeneity in the representation and modeling of QoS information by each service provider, as we have described earlier in the background. With this heterogeneity in QoS representation models, QoS brokers should provide a common ontology-based QoS model and the mappings from the various models to this common model. Furthermore, the interaction model, described in previous sections, provides the basis for the development of an API that all components of the framework can use to interact with each other. Heterogeneity of the APIs offered by various QoS brokers and service providers is one of the challenges of the approach.

5. CONCLUSION AND FUTURE WORK

As a result of the emergent demand for QoS-aware services, service providers are increasingly using SOA and the Web services technology to implement services that can ensure several QoS levels. In this paper, we have presented a novel framework for QoS-aware service provisioning. The framework relies on QoS brokers, to mediate between clients and service providers, and a QoS Notification broker, to handle the notifications on the changes in the QoS offerings of service providers, using a publish/subscribe model.

We have described the model of interactions among the components of the framework, and a multi-attributes decision algorithm for the ranking and selection of appropriate service providers that can meet the client QoS requirements.

As a future work, we are planning to investigate more on the issue of a common ontology-based model for QoS representation that all components of the framework can use; and then, describe the mappings from the various QoS representation models described in the literature to that common model. Moreover, we intend to build a prototype of the framework together with some real scenarios for QoS-aware service provisioning.

6. REFERENCES

[1] Catania,N., Kumar,P., Murray,B., PourhedariH., VambenepeW., and WursterK., 2003. "Web Services Management Framework, Version 2.0," Hewlett Packard, http://xml.coverpages.org/WSMF-Overview.pdf

[2] Tosic,V., Pagurek,B., and Patel,K., 2003. "WSOL – A Language for the Formal Specification of Classes of Service for Web Services," In Proceedings of The 2003 International Conference on Web Services (ICWS'03), CSREA Press, pp. 375-381.

[3] Yeom, G. and Min,D., 2005. "Design and Implementation of Web Services QoS Broker," In Proceedings of The International Conference on Next Generation Web Services Practices (NWeSP 2005), pp. 459- 461.

[4] Tao,Y. and Lin,K.J., 2005. "A Broker-based Framework for QoS-aware Web Service Composition," In Proceedings of The 2005 IEEE International Conference on eTechnology, e-Commerce and e-Service (EEE'05), pp. 22-29.

[5] Zuquim, G.D. and Felgar de Toledo,M.B., 2006. "A Web Service Architecture Providing QoS Management," In Proceeding of The Fourth Latin American Web Congress (LA-WEB'06), pp. 189-198.

[6] Dan,A. et al., 2004. “Web services on demand: WSLAdriven automated management,” IBM Systems Journal, 43(1), pp. 136-158.

[7] Menascé,D.A., 2002. “QoS Issues in Web Services,” IEEE Internet Computing, 6(6), pp. 72–75.

[8] D’Ambrogio,A., 2006. "A model-driven wsdl extension for describing the qos of web services," in Proceedings of the International Conference on Web Services (ICWS’06).

[9] Kang,Y.H., 2007. "Extended Model Design for Quality Factor Based Web Service Management," Future Generation Communication and Networking (FGCN 2007), Vol. 2.

[10] Lee,Y. and Yeom,G., 2007. “A Quality Chain Modeling Methodology for Ternary Web Services Quality View,” In Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA '07), pp. 91-97.

[11] Lo,C.C., Cheng,D.Y., Lin,P.C.,and Chao,K.M., 2008. "A study on representation of QoS in UDDI for web services composition," In International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2008), pp. 423-428.

[12] Blum,A. and Carter,F., 2004. "Representing Web Services Management Information in UDDI".

[13] Xu,Z., Martin,P., Powley,W. and Zulkernine,F., 2007. "Reputation-enhanced qos-based web services discovery," In IEEE International Conference on Web Services (ICWS 2007), pp. 249-256.

[14] Shaikh Ali,A., Rana,O.F., Al-Ali,R., and Walker,D.W., 2003. “UDDIe: an Extended Registry for Web Services,” In Proceedings of The IEEE Symposium on Applications and the Internet Workshops, pp. 85 – 89.

[15] Tian,M., Gramm,A., Naumowicz,T., Ritter,H., and Schiller,J., 2003. “A Concept for QoS Integration in Web Services”, In Proceedings of the First IEEE Web Services Quality Workshop.

[16] Yu,T. and Lin,K.J., 2004. “The Design of QoS Broker Algorithms for QoS-Capable Web Services,” In Proceedings of the IEEE International Conference on eTechnology, e-Commerce and e-Service (EEE'04), Vol. 00, pp. 17-24.

[17] Badidi,E. andEsmahi,L., 2011. “A Scalable Framework for Policy-based QoS Management in SOA Environments,” Journal of Software, Academy Publisher, 6(4) pp. 544-553.

[18] Zuquim,G.D. and Felgar de Toledo,M.B., 2006. “A Web Service Architecture Providing QoS Management,” In Proceeding of The Fourth Latin American Web Congress (LA-WEB'06), pp. 189-198.

[19] Chaari,S., Badr,Y., and Biennier,F., 2008. “Enhancing Web Service Selection by QoS-based Ontology and WSPolicy,” In Proceedings of The ACM Symposium on Applied Computing (SAC 2008), pp. 2426-2431.

[20] Bajaj,S., et al., 2007. “Web Services Policy 1.5 Framework,” W3C Candidate Recommendation 28 February 2007. http://www.w3.org/TR/2007/CR-ws-policy-20070228/

[21] W3C, “Web Services Policy Attachment,” http://www.w3.org/Submission/WS-PolicyAttachment".

[22] Trastour,D., Bartolini,C., and Castillo,J.G., 2001. “A semantic Web approach to service description for matchmaking of services,” In Proceedings of the International Semantic Web Working Symposium (SWWS).

[23] Maximilien,E.M. and Singh,M.P., 2004. “A Framework and Ontology for Dynamic Web Services Selection,” IEEE Internet Computing, 8(5), pp. 84–93.