Analyses of QoS-Aware Web Services

Stephen J. H. Yang
Dept. of Computer Science and Information Engineering, National Central University, Taiwan
jhyang@csie.ncu.edu.tw

Blue C. W. Lan
Dept. of Computer Science and Information Engineering, National Central University, Taiwan
lancw@csie.ncu.edu.tw

Jen-Yao Chung
IBM T. J. Watson Research Center, USA
jychung@us.ibm.com

Abstract

Quality-of-Service (QoS) in Web services considers a service's non-functional characteristics during service advertisement, discovery and composition. In this paper, we analyze the requirements of QoS-aware Web services and present the corresponding findings as well. We firstly propose a common set of QoS attributes with formal definitions to encourage the creation of a general QoS model for Web services. Based on the attributes, we define the process of QoS-aware service discovery and two alternative matching criteria namely absolute and relative matchmaking are discussed to enable flexible service selection. Finally, we present the aggregative effects of QoS attributes according to the formal semantics of different workflow patterns to help service consumer perform QoS-aware service composition.

Keyword: QoS, Web services, non-functional

1. Introductions

The widespread Internet accessibility and World Wide Web popularity make today's e-commerce more complicated than it was before. How to deliver application functionality in a timely, flexible and trustworthy manner has become a great challenge now. Web services emerged along with XML technologies to help IT developers deal with the heterogeneity among software applications. By utilizing standards-based Web services model, it is able to rapidly design, implement and deliver desired functionality. Due to the characteristics of low entry cost, low barriers and standard approaches derived from Web, XML and Internet technologies, Web services are viewed as an important enabling technology for the next-generation e-commerce and Gartner Inc. [1] predicts that there are more than 60% of businesses will adopt Web services by 2008. The growing popularity of Web services has resulted in an ever-evolving specification stack as illustrated in Figure 1. Numerous specifications are proposed for different purposes and the abundance of overlapping specifications has led Web services developments to an acronym hell where specifications appear without clear added-value. Besides, the majority of specifications highlight the way of delivering functionalities and few are dedicated to Quality-of-Service (QoS) concerns of Web services.

QoS concerns of Web services concentrate the attention on the fulfillment of non-functional attributes such as reliability, availability, security and response time. Because of the loosely-coupled and dynamic natures, the adoption of Web services may suffer from several uncertainties, for example, how to ensure that the service will perform reliably? Is the found service available while it is needed? How to keep confidentiality of transmitted data? And how long the service takes for the execution? In order to advance the prevalence of Web services without uncertainties, it is critical to develop Web services in a QoS-aware or trustworthy manner [2]. In this paper, we propose developing QoS-aware Web services through three stages including (1) creation of a general QoS model of Web services: QoS concern in Web services should be an end-to-end issue. Service provider and consumer should get a consensus of the definitions of non-functional attributes. Then it is possible for service provider and consumer to describe the QoS characteristics and requirements without ambiguities. (2) QoS-aware service discovery: In additional to functional matchmaking, another estimation algorithms or methods are required to determine whether services are satisfied with consumer's QoS requirements. (3) QoS-aware service composition: In contrast to individual QoS-aware service discovery, service consumer needs to select constituent services in a service composition with a global view of QoS requirements. Based on different workflow patterns, the overall QoS performance of a composite service will be evaluated aggregately. The contributions of this paper focus on analyzing the requirements of developing

QoS-aware Web services through proposed three stages. Both researchers and practitioners can benefit from the providing guidelines as well. The remaining of the paper is organized as follows. Requirements of creating a general QoS model of Web services are presented in section 2. Analyses of QoS-aware service discovery and composition are specified in section 3 and 4 respectively. Finally, concluding remarks are described in section 5.

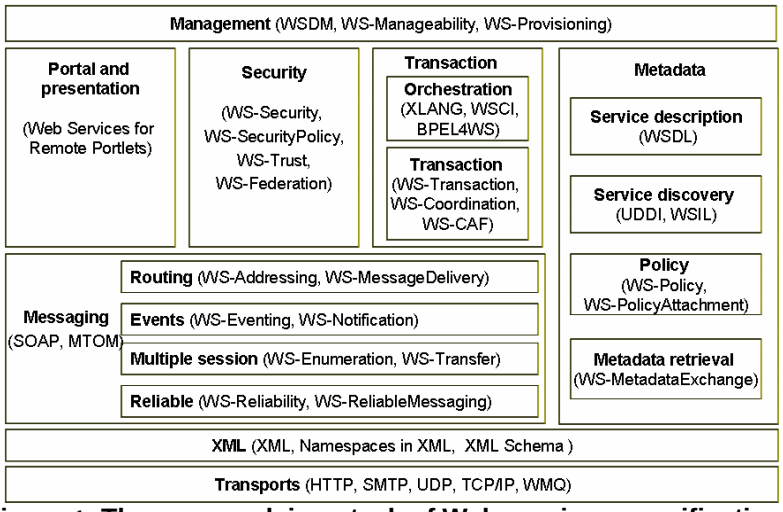

Figure 1. The ever-evolving stack of Web services specifications

2. QoS model of Web services

The concept of quality or Quality-of-Service (QoS) usually has different definitions from divergent perspectives. For example, *"Quality of Service refers to the probability of the telecommunication network meeting a given traffic contract"* [3], *"The degree to which a system, component or process meets specified requirements"* and *"The degree to which a system, component or process meets customer or user needs or expectations"* [4]. Based on the definitions, we define QoS-aware Web services in this paper as the services which are aware of service consumer's functional and non-functional requirements during service advertisement, discovery and composition.

In order to describe QoS-aware Web services universally, a general service description model is required for service advertisement. For functional description, Web services description language (WSDL) [5] has provided a standard model to describe service's functionality by separating the abstract representations of service's input and output messages from the concrete descriptions of end point's bindings. However, there is no general QoS model to capture service's non-functional characteristics until now. In [6], multiple dimensions of quality are discussed including performance, features, reliability, conformance, durability, serviceability, aesthetics and perceived quality. Both subjective concerns such as the image of brand name and objectively measurable attributes such as mean time to first failure (MTFF) are involved. By considering the features of Web services, various QoS attributes including availability, security, response time, throughput, cost, reliability, fidelity and trust etc. have been defined [7; 8; 9]. But the definitions of the attributes are informal such that service consumer and provider may interpret the attributes ambiguously. In order to help in creating a general QoS description model, we synthesize fore-mentioned works to propose a common set of QoS attributes as illustrated in Table 1 and the formal definitions of each QoS attribute are specified in the following.

Table 1. A common set of QoS attributes

Dimensions	Attributes
Performance	Response time
	Throughput
Dependability	Reliability
	Availability
Cost	Price
Security	Authentication
	Confidentiality
	Integrity
	Non-repudiation

(1) **Response time:** Temporal issue is a basic performance concern in Web services and response time is a typical performance attribute that refers to the elapsed time between the end of issuing a request to a service and the beginning of the service's response. The evaluation of a service's response time for a request R can be represented as shown below.

Response time(R) = Execution time(R) + Waiting time(R)

The execution time is the duration of performing the service functionality and the waiting time is the amount of time for all possible mediate events such as message transmissions between service consumer and provider. However, the evaluation of response time is controversial due to the uncertain network fluctuations. From service consumer perspective, it is meaningful to take response time as the duration starting at issuing the request and ending with the receipt of service's response. But from service provider perspective, response time is taken as same as execution time and it does not include all possible mediate events, which are seen as incontrollable variables during service execution. The gap comes from the fact that service provider has no way to precisely describe the response time of offered service if waiting time is considered. In order to minimize the gap, a flexible description method is required to balance the two viewpoints.

(2) **Throughput:** It is critical for service consumer to know the amount of work that a service can perform in a given period of time, for example, number of requests per second. In some scenarios, e.g. airline booking, intensive inquiries are often dumped in a short time so it is important for service consumer to ensure whether service's throughput can fulfill the volume of anticipated requests. The throughput of a service S can be represented as follows.

Throughput(S) = Number of requests / per unit-of-time

According to service's granularity, the unit-of-time may vary from mini-second to minute. Similarly, a flexible description method is required to deal with different granularity.

(3) **Reliability:** One of the most significant QoS concerns of Web services is service's reliability which refers to the ability of successfully performing functions for a specified period of time. The ability is able to be quantitatively defined by the probability of if a service can deliver the functionality successfully. Reliability of a service S can be represented by the failure rate as shown below.

Reliability(S) = 1 – Failure rate(S)

The failure rate of a service could be measured by the ratio of execution time and mean time between failures (MTBF). Service provider may need to carry out plenty of simulations to obtain accurate probabilistic value of offered service's reliability.

(4) **Availability:** The degree to which a service is operational and accessible when it is required for use determines the service's availability. The availability of a service S can be defined by the proportion of the service's uptime to downtime as follows.

Availability(S) = Uptime(S) / Uptime(S) + Downtime(S)

The uptime and downtime of a service can be measured by the mean time between failures (MTBF) and mean time to recovery (MTTR) respectively.

(5) **Price:** The expense for a service execution is always associated with the value of service's functionality. The higher rates a service takes the more complicated functions the service provides. The price for executing a service S can be represented as follows.

Price(S) = Execution fee(S) / per request

Generally, both functional and non-functional performance fulfillment of services with charge should be guaranteed to service consumer by service level agreements (SLA), which legally bind contracts to reach the promises during service execution.

(6) **Authentication:** As Web services emerge progressively on the horizon, how to benefit from the adoption of this new technology without compromising security concerns is crucial to its extensive use in the near future. In terms of Web services, authentication is the capability of distinguishing a man from a fraud remotely. In order to stop an intruder from masquerading as service provider, it should be able for service consumer to identify service provider's identity of. The authentication capability of a service S and the corresponding service provider P can be represented as shown below.

Authentication(S, P) = Security token(S, P)

The security token is a collection of claims that are declarations made by service provider to specify his name, identity and his supportive authentication methods.

(7) **Confidentiality:** How to keep eavesdropper from reading transmitted data is another important security concern in Web services. Enterprises may utilize Web services to carry out business transactions and sensitive business data may be exposed to anyone who can access Internet. Enterprises as service consumer will not adopt Web services until the confidentiality of transmitted data is promised. The capability of confidentiality guarantee offered by a service S can be represented as follows.

Confidentiality(S) = Security token(S)

The security token encompasses all supportive encryption and decryption methods.

(8) **Integrity:** Considering that many significant data may be carried by Web services, it should be able for the receiver of a message to verify that the message has not been modified in transit. In other words, an intruder should not be able to substitute a fake message for a legitimate one. The integrity promise of a service S can be represented as follows.

Integrity(S) = Security token(S)

The security token specifies a collection of claims that demonstrate the service's capability of integrity promise.

(9) **Non-repudiation:** Since Web services are seen as an

important enabling technology for next-generation e-business, all exchanged messages between service consumer and provider are a kind of agreement. A sender should not be able to falsely deny later that he sent a message. The capability of non-repudiation warranty provided by a service S can be represented as follows.

Non-repudiation(S) = Security token(S)

The security token includes all supportive methods for non-repudiation warranty.

The fore-mentioned attributes present a common QoS view in Web services and they are helpful to the creation of a general QoS description model. However, there are still some controversies over the definitions of QoS attributes, e.g. the calculation of response time and different charge styles for a service execution etc. How to design a general, flexible and extensible QoS description model has become a pressing issue toward the developments of QoS-aware Web services.

3. QoS-aware service discovery

For matching Web services with service consumer's functional requirements, UDDI [10] offers consumer a systematical way to find out desired services through centralized service registry. There are three kinds of information about a registered Web service, i.e. white pages include information of name and contact details, yellow pages provide a categorization upon business and service types and green pages specify technical data of the services. Based on these three encoding information, UDDI can support keyword or directory-based service discovery. However, such service selection process is suitable for text based attributes e.g. service name and provider name. It is insufficient to handling numeric QoS attributes, for example, response time and reliability etc.

According to different data types of service attributes, the service selection process of QoS-aware service discovery is carried out within 3 steps as illustrated in Figure 2.

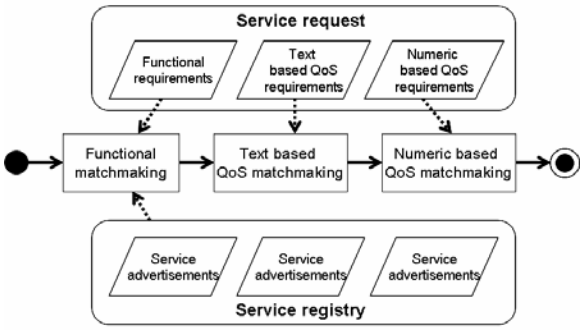

Figure 2. Process of QoS-aware service discovery

(1) Functional matchmaking: Based on service consumer's functional requirements, a set of services is selected through UDDI mechanism. (2) Text based QoS matchmaking: The set of services from step 1 is further matched with text based QoS attributes by UDDI mechanism. (3) Numeric based QoS matchmaking: By comparing the numeric values of service attribute and QoS requirement, a set of services which fulfill the request is selected finally. Based on the process, we propose two alternative matching criteria namely absolute and relative matching for service consumer as shown in Table 2.

Table 2. Two alternative matching criteria

Criterion / Data type	Absolute matching	Relative matching
Text	UDDI	Enhanced UDDI
Numeric	Arithmetic subtraction	multiple criteria decision making (MCDM) with weighted sum model (WSM)

(1) **Absolute matching for text:** We apply keyword or directory-based service discovery as UDDI to perform exact match against text based attributes including functional characteristics and authentication, confidentiality, integrity and non-repudiation.

(2) **Absolute matching for numeric:** In terms of numeric attribute, we define a service is satisfied with a request under absolute matching if the value of service's positive/negative attribute is greater/lower or equal to the request value as shown below.

q.value >= r.value for positive QoS attributes

q.value <= r.value for negative QoS attributes

Positive attribute indicates that the higher the attribute value is the better the quality is, e.g. throughput, reliability and availability. Inversely, negative attributes including response time and price signify that the higher the attribute value is the worse the quality is.

(3) **Relative matching for text:** In contrast to the absolute matching, the relative matchmaking provides a flexible keyword or directory-based service discovery with wild character to carry out partial match. Wild characters can be put in any places to express more general queries.

(4) **Relative matching for numeric:** Instead of exactly specifying required performance of attributes, relative matching allows service consumer allocating weighted values for each attribute and setting a threshold score to issue a loose request. By using the multiple criteria decision making (MCDM) technique with weighted sum model (WSM), the relative matching for numeric is performed within 2 steps as shown below [11].

(i) **Normalization of attribute value:** The value of

each numeric attribute q of a candidate service is normalized with the following equations.

$$q.value = \begin{cases} \dfrac{q.value - q.\min}{q.\max - q.\min} & if \quad q.\max - q.\min \neq 0 \\ 1 & if \quad q.\max - q.\min = 0 \end{cases} \qquad Eq(1)$$

$$q.value = \begin{cases} \dfrac{q.\max - q.value}{q.\max - q.\min} & if \quad q.\max - q.\min \neq 0 \\ 1 & if \quad q.\max - q.\min = 0 \end{cases} \qquad Eq(2)$$

Positive and negative attributes are normalized by Eq(1) and Eq(2) respectively. Besides, q.max and q.min are the maximal and minimum value of the attribute among all candidate services.

(ii) **Weighting and sum of each numeric attributes:** Each normalized numeric attributes q of a candidate service s multiplies the corresponding weight w given by service consumer will generate an overall evaluation score of the service as shown below.

$$Score(s) = \sum q.value * w$$

Services whose evaluation scores are greater or equal to the threshold score given by service consumer will be selected.

The current UDDI standard has offered the partial match functionality with wild characters but it still focuses on functional matchmaking only. A QoS-aware service discovery solution should not only take care of various data types of QoS attributes but also be able to provide flexible service selection methods accordingly.

4. QoS-aware service composition

Service composition is the process of creating new functionalities by aggregating several independent services. In the process, a lot of workflow patterns are applied to shape these services into a new composite service with added value functionalities. From service consumer's perspective, the QoS performance of a composite service is perceived aggregately from the performance of its constituent service. Thus service selection for a QoS-aware service composition should be carried out with a global view of QoS attributes [12]. In general, the service selection for QoS-aware service composition depends on numeric attribute only and there is no aggregative effect of text based attribute. For example, the performance of two interrelated services' authentication capability is always perceived consistently regardless of their composing patterns. The process of QoS-aware service composition is carried out through 3 steps as illustrated in Figure 3.

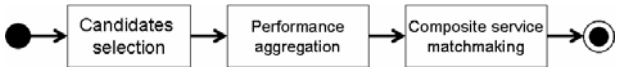

Figure 3. Process of QoS-aware service composition

(1) **Candidates selection:** Numerous candidate services for a service composition are retrieved by functional and text based QoS matchmaking as specified previously.

(2) **Performance aggregation:** Based on the formal semantics of workflow patterns and the definitions of QoS attributes, the aggregative performance of a composite service can be derived from the performance of its constituent services. Figure 4 illustrates some useful workflow patterns with Petri nets [13] and the corresponding aggregative effect of numeric attributes is shown in Table 3.

(3) **Composite service matchmaking:** According to the performance aggregation, a set of services for a service composition can be taken as an atomic service and service consumer can apply numeric based QoS matchmaking as specified in Section 3 to determine the best set of services.

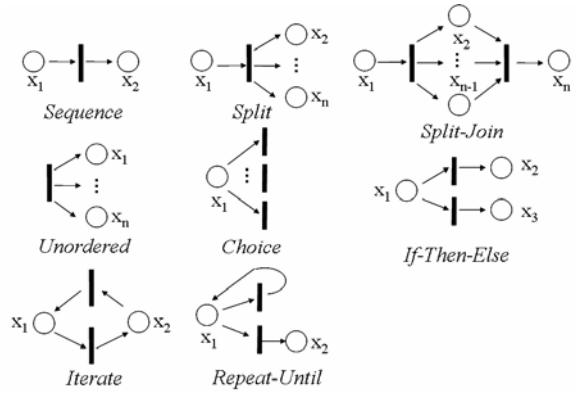

Figure 4. Workflow patterns with Petri nets

Table 3. Aggregative effect of numeric QoS attributes

Patterns \ Attributes	Response time	Throughput	Reliability	Availability	Price
Sequence	$x_1 + x_2$	$\min\{x_1, x_2\}$	$x_1 * x_2$	$x_1 * x_2$	$x_1 + x_2$
Split	$x_1 + \max\{x_2,\ldots x_n\}$	$\min\{x_1, \ldots x_n\}$	$x_1 * \ldots * x_n$	$x_1 * \ldots * x_n$	$x_1 + \ldots + x_n$
Split-Join	$x_1 + \max\{x_2,\ldots x_{n-1}\} + x_n$	$\min\{x_1, \ldots x_n\}$	$x_1 * \ldots * x_n$	$x_1 * \ldots * x_n$	$x_1 + \ldots + x_n$
Unordered	$\max\{x_1,\ldots x_n\}$	$\min\{x_1, \ldots x_n\}$	$x_1 * \ldots * x_n$	$x_1 * \ldots * x_n$	$x_1 + \ldots + x_n$

Choice	x_1	x_1	x_1	x_1	x_1
If-Then-Else	$x_1 + \max\{x_2, x_3\}$	$\min\{x_1, x_2, x_3\}$	$x_1 * \min\{x_2, x_3\}$	$x_1 * \min\{x_2, x_3\}$	$x_1 + \max\{x_2, x_3\}$
Iterate	$n * (x_1 + x_2)$	$\min\{x_1, x_2\}$	$(x_1 * x_2)^n$	$(x_1 * x_2)^n$	$n * (x_1 + x_2)$
Repeat-Until	$n * x_1 + x_2$	$\min\{x_1, x_2\}$	${x_1}^n * x_2$	${x_1}^n * x_2$	$n * x_1 + x_2$

Jaeger et al [14] also identify the aggregation of numerical QoS dimensions for some workflow patterns but their aggregation may be arguable due to the missing formal semantics of the workflow patterns. On the other hand, Zeng et al [15] discussed the computational complexity problem of choosing the best set of services. The volume of candidate sets of services for a service composition is proportional to the amount of available candidate services and thus the computational complexity of a brute-force estimation method will be exponential. Hence, a solution of QoS-aware service composition should define formal semantics of different workflow patterns and provide the corresponding aggregative effects as well. Besides, how to help service consumer select the best set of services for a service composition with low computational complexity should also be considered.

5. Conclusions

The development of QoS-aware Web services is a popular research issue as it is seen as the foundation toward trustworthy Web services. The promise of providing services with certain QoS performance will make service consumer be more confident of adopting Web services for critical tasks. In order to benefit both service provider and service consumer, a general QoS model of Web services is required. The model should balance different viewpoints from the two parties and provide formal definitions of each QoS attribute such that there is no ambiguity in interpreting attributes. Based on the model, QoS-aware service discovery should provide flexible service selection methods. According to the characteristics of different attributes, distinct matchmakings can be applied to service consumer's requirements correspondingly. For complicated composite services, QoS-aware service composition should take care of various workflow patterns. Based on the formal semantics of different patterns, the corresponding aggregative effects of each QoS attribute can be derived from constituent services and the selection of candidate sets of services for a service composition can be done by QoS-aware service discovery mechanisms. In the near future, we will focus on verifying the promise of providing QoS-aware Web services. The guarantee of QoS performance should be proved during service execution. The challenges of service monitoring and failure recovery will be worth studying.

Acknowledgement

This work is supported by National Science Council, Taiwan under grants NSC 94-2524-S-008-001.

References

[1] M. Pezzini. (2003, Oct.). Composite Applications Head Toward the Mainstream. Gartner, Inc. Stamford, U.S.A. [Online]. Available: http://www.gartner.com

[2] J. Zhang, "Trustworthy Web Services: Actions for Now," *IEEE IT Professional*, vol. 7, issue 1, Jan. / Feb. 2005, pp. 32-36.

[3] Wikipedia. (2006, Aug.). Quality of Service. [Online]. Available : http://en.wikipedia.org/wiki/Quality of_service

[4] F. Jay and R. Mayer, "IEEE Standard Glossary of Software Engineering Terminology," IEEE Std 610.12-1990, Sep. 28, 1990.

[5] R. Chinnici et al. (ed.) (2006, Mar.). Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. WWW Consortium. [Online]. Available: http://www.w3.org/TR/wsdl20/

[6] D. A. Garvin, "Managing quality: The strategic and competitive edge," New York: Free Press, 1988, pp.49-68.

[7] D. A. Menasce, "QoS Issues in Web Services," *IEEE Internet Computing*, vol. 6, issue 6, Nov. / Dec. 2002, pp. 72-75.

[8] J. Cardoso, J. Miller, A. Sheth and J. Arnold, "Modeling Quality of Service for Workflows and Web Service Processes," LSDIS lab, Computer Science, University of Georgia, Tech. Rep. #02-002, Dec. 2002.

[9] J. O'sullivan, D. Edmond and A. T. Hofstede, "What's in a service? Towards Accurate Description of Non-Functional Service Properties," *Kluwer Academic Distributed and Parallel Databases*, vol. 12, 2002, pp. 117-133.

[10] L. Clement et al. (2004, Oct.). UDDI Version 3.0.2. *OASIS*. [Online]. Available: http://uddi.org/pubs/uddi_v3.htm.

[11] C.L. Hwang and K. Yoon, "Multiple attribute decision making: Methods and applications," *Springer-Verlag*, 1981.

[12] D. A. Menasce, "Composing Web Services: A QoS View," *IEEE Internet Computing*, vol. 8, issue 6, Nov. / Dec. 2004, pp. 88-90.

[13] S.J.H. Yang, B.C.W. Lan and J.Y. Chung, "A New Approach for Context Aware SOA," in Proc. of IEEE International conference on e-Technology, e-Commerce and e-Service (EEE), 2005, pp. 438-443.

[14] M.C. Jaeger, G. Rojec-Goldmann and G. Muhl, "QoS aggregation for Web service composition using workflow patterns," in Proc. of IEEE International conference on Enterprise Distributed Object Computing (EDOC), 2004, pp. 149-159.

[15] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Transaction on Software Engineering*, vol. 30, no. 5, May 2004, pp. 311-327.