Service Level Agreement-Based QoS Analysis for Web Services Discovery and Composition

Stephen J.H. Yang

Department of Computer Science and Information Engineering,
National Central University,
No.300, Jhongda Rd., Jhongli City, Taoyuan County 32001, Taiwan (R.O.C.)
Phone: 886-3-4227151 ext. 35308
E-mail: jhyang@csie.ncu.edu.tw

Jia Zhang

Department of Computer Science,
Northern Illinois University
1425 W. Lincoln Hwy. DeKalb, IL 60115-2825, USA
Phone:1-312-7182468
E-mail: jiazhang@cs.niu.edu

Blue C.W. Lan

Department of Computer Science and Information Engineering,
National Central University,
No.300, Jhongda Rd., Jhongli City, Taoyuan County 32001, Taiwan (R.O.C.)
Phone: 886-3-4227151 ext. 35326
E-mail: lancw@csie.ncu.edu.tw

Abstract: Quality-of-Service (QoS) in Web services considers a service's non-functional characteristics during service specification, discovery, and composition. In order to encourage the development of QoS-aware Web services, we first develop a QoS-aware model, which contains a common set of QoS attributes including response time, throughput, reliability, availability and price etc. Then, based on the attributes, two alternative service selection methods, namely absolute and relative matchmaking, are presented. Finally, according to the formal semantics of different workflow patterns, we utilize the aggregative effects of QoS attributes to help service consumers perform QoS-aware service composition.

Keywords: QoS, Web services, service specification, service discovery, service composition, service level agreement

About The Authors

***Stephen J.H. Yang*, Ph.D.**, is an Associate Professor of the Department of Computer Science and Information Engineering, National Central University, Taiwan. He was the co-founders and the CEO of T5 Corp, a company providing XML-based Web services. Dr. Yang has published 2 books and over 100 technical papers in the areas of software engineering and knowledge engineering. He severed as the Program Co-Chairs of IEEE MSE2003 and CAUL2006. His research interests include software engineering, knowledge engineering, semantic Web, context aware ubiquitous computing, peer to peer computing, and mobile multimedia. Dr. Yang received his PhD degree in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1995. He is a member of IEEE and ACM.

***Jia Zhang*, Ph.D.**, is an Assistant Professor of Department of Computer Science at Northern Illinois University. She is also a Guest Scientist of National Institute of Standards and Technology (NIST). Her current research interests center around services computing. Zhang has published over 60 technical papers in journals, book chapters, and conference proceedings. She is an Associate Editor of the International Journal of Web Services Research (JWSR), and the Program Vice Chair of IEEE International Conference on Web Services (ICWS 2006). Zhang received a Ph.D. in Computer Science from University of Illinois at Chicago in 2000. She is a member of the IEEE and ACM.

Blue Ci-Wei Lan is a Ph.D. student at the Department of Computer Science and Information Engineering, National Central University, Taiwan. He received the B.S. and M.S. degree in Mathematics and CSIE from NCU in 1999 and 2001 respectively. His research interests are semantic Web services, intelligent software agent and e-business solutions.

1. Introduction

The widespread Internet accessibility and World Wide Web popularity make today's e-commerce more complicated than it was before. How to deliver Web applications in a timely, flexible, and trustworthy manner has become a great challenge for enterprises to perform Business-to-Consumer (B2C) and Business-to-Business (B2B) transactions. 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 and thus enterprises will be more responsive, efficiency and cost-effective in terms of adapting to the ever-changing business environment. 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. Gartner Inc. (Pezzini, 2003) predicted that 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 the purpose of service description, discovery, orchestration, presentation and management etc. However, 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 functionality of Web

services delivery, few of them are dedicated to Quality-of-Service (QoS), especially the non-functional concerns of Web services.

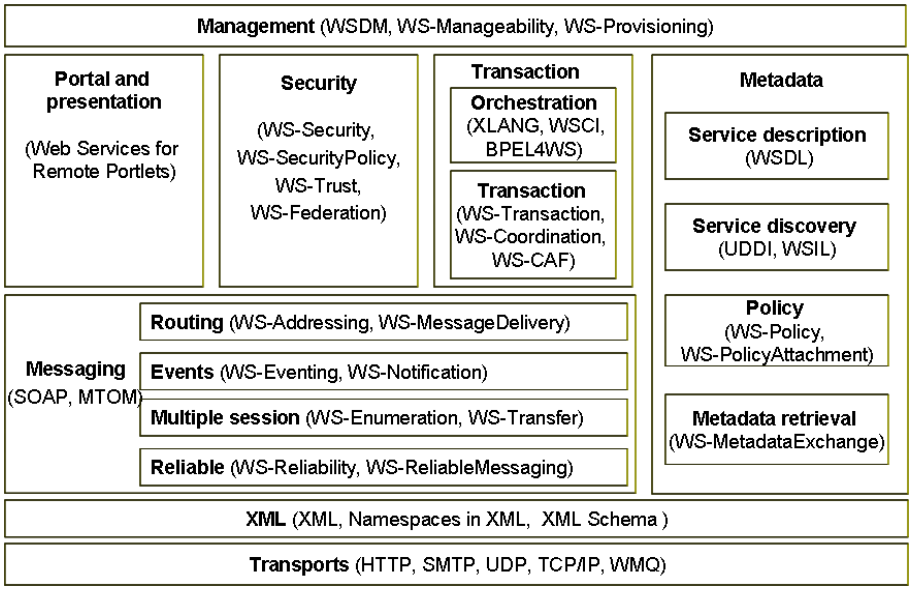

Figure 1. The ever-evolving stack of Web services specifications

Web services' QoS concerns concentrate 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 a service will perform reliably? Will the found service be available while it is needed? How to keep confidentiality of transmitted data? And how long is a service's execution time? 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 (Zhang, 2005).

The major contribution of this paper is to present a QoS-aware model for developing Web services through three stages including

(1) QoS specification of Web services: Web services' QoS concern should be an end-to-end issue. When service providers and consumers reach agreements of the definitions of non-functional attributes, then it is possible for service providers and consumers to describe 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 non-functional QoS requirements.
(3) QoS-aware service composition: In contrast to individual QoS-aware service discovery, service consumers need to select constituent services in service composition with a global view of QoS requirements. Based on workflow patterns, the overall QoS performance of a composite service will be evaluated aggregately.

The remainder of the paper is organized as follows. Section 2 will address the requirements of creating a general QoS model for Web services development, and

Section 3 will address service level agreement and QoS deduction. QoS-aware service discovery and composition and monitoring will be presented in Section 4. Finally, concluding remarks are described in Section 5.

2. QoS-aware Service Specification

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"* (Wikipedia, 2006), *"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"* (Jay and Mayer, 1990). 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, composition, and execution.

Standard Web service description language such as Web services description language (WSDL) (Chinnici et al., 2006) provides a 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. Similarly, a general QoS model will be needed for the developments of QoS-aware Web services. In (Garvin, 1988), multiple dimensions of quality have been discussed including performance, features, reliability, conformance, durability, serviceability, aesthetics and perceived quality. Both subjective concerns such as image of brand name and objectively measurable attributes such as mean time to first failure (MTFF) are involved. For considering the characteristics of Web services, various QoS attributes which are specifically defined for Web services such as availability, security, response time, throughput, cost, reliability, fidelity and trust etc can be found in (Menasce, 2002; Cardoso et al, 2002; O'sullivan, Edmond and Hofstede, 2002). In order to facilitate the creation of a general QoS model for Web service development, we have aggregated fore-mentioned work and present a common set of QoS dimensions and attributes in Web services as illustrated in Table 1.

Table 1. QoS dimensions and attributes in Web services

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

(1) **Response time:** Response time is a typical measurable performance attribute that refers to the elapsed time between the initiation of a service request and the completion of the service's response. The evaluation of response time usually consists of execution time and waiting time. 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 service functionality. The waiting time is the amount of time for all possible mediate events such as message transmissions between service consumers and providers. However, the evaluation of response time is controversial due to the uncertainty of network fluctuations. From service consumer perspective, it is meaningful to consider response time as the duration starting from the issue of a request to the end of receipt of a service's response. But from service provider perspective, response time is considered as same as execution time of a service, so it does not include all possible mediate events, which are seen as incontrollable variables during service execution. The gap is because of the fact that service providers cannot precisely describe the waiting time of a service execution. In order to minimize the gap, a flexible description method is required to balance the two viewpoints.

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

$$Throughput\ (S) = Number\ of\ requests\ /\ per\ unit\text{-}of\text{-}time$$

According to the service's granularity, the unit-of-time may vary from mini-second to minute. As well as response time, a flexible description method is required to adapt throughput descriptions to different services.

(3) **Reliability:** One of the most significant QoS concerns of Web services is reliability, which refers to the ability of a service to perform its offered functions for a specified period of time. The ability can be quantitatively perceived by the probability 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 can be measured by the ratio of execution time and mean time between failures (MTBF). Service providers may need to carry out plenty of simulations for obtaining accurate value of service's reliability.

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

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

The uptime of a service can be measured by the mean time between failures (MTBF) and the downtime can be measured by the mean time to recovery (MTTR). Similarly, a lot of simulations should be performed to get precise value of service's availability.

(5) **Price:** The expense regarding a service execution is associated with the value of service's functionality. The higher price a service costs, 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, for functional and non-functional performance of services which are not free of charge they should provide guarantee to service consumers with service level agreements (SLA). SLA legally bind contracts to reach the promises during service execution.

(6) **Authentication:** As Web services emerge progressively, how to benefit from the adoption of this new technology without compromising security concerns will be crucial to its extensive use in the near future. In terms of Web services, authentication is the capability to distinguish a man from a fraud remotely. In order to stop an intruder from masquerading as a service provider, service consumers should be enabled to identify the service provider. The authentication of a service, *S*, and the corresponding service provider, *P*, can be represented as shown follows.

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

The security token is a collection of claims that are declarations made by service providers to specify their names, identities, and their supportive authentication methods.

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

$$Confidentiality(S) = Security\ token(S)$$

The security token should encompass 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 during transmission. 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 consumers and providers are a kind of agreement. A sender should not falsely deny that he/she has sent a message. The capability of non-repudiation warranty provided by a service, *S*, can be represented as follows.

$$Non\text{-}repudiation(S) = Security\ token(S)$$

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 model of Web services. 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. Besides, even though numerous specifications have been proposed for different purposes as illustrated in Figure 1, none of them can provide a uniform syntactic description model for non-functional attributes of Web services as WSDL does for service's functional characteristics. How to design a general, flexible and extensible QoS model has become a demanding requirement toward the developments of QoS-aware Web services.

3. SLA-Based QoS Deduction

3.1 Service Level Agreement (SLA)

As we have defined the QoS attributes of Web services, the next question is how to find out the QoS attributes of a Web service from the Internet. Although the ideal way to obtain the QoS data of a Web service is through testing, as the initial filtering, we can consider the published QoS features of services, assuming that the published information is no worse than the actual features in common sense. As an *ad hoc* industry standard, Service Level Agreement (SLA) is widely used to define a formal contract associated with a Web service between a service consumer and a service provider, aiming at specifying quantifiable issues under specific contexts based upon mutual understandings and expectations (WS-Agreement, 2005). SLA can thus be used to define any service-related issue, including QoS factors.

To date, several SLA specifications and proposals are available. Among them, two catched most attentions: Web Services Agreement Specification (WS-Agreement) from Global Grid Forum (WS-Agreement, 2005) and Web Service Level Agreement (WSLA) from IBM (IBM, 2003). They both define their XML-based languages and protocols for service providers to advertise Web services. In our research, we deduce QoS attributes of a Web service from its service provider SLA documents in the WS-Agreement format. WS-Agreement is proposed by Global Grid Forum (GGF) (GGF, 2005), which is a community consisting of thousands of individuals in both academia and industry around the world. It should be noted that our approach of SLA-oriented QoS elicitation is not limited to WS-Agreement; instead, it can be easily applied to any XML-based SLA standards (e.g., WSLA) with limited changes.

A WS-Agreement-based specification generally contains three parts: (1) the schema of the agreement, (2) the schema of the agreement template, and (3) agreement-specific life-cycle management port types and operations. Such an agreement can be used to define service level state-dependent requirements as expressions of resource availabilities (e.g., memory, CPU, and disk space) and QoS attributes (e.g., response time).

3.2 SLA document-based QoS Analysis

Our fundamental idea is that the definitions in an SLA document from a service provider can be used to facilitate service discovery and service composition. If a service provider provides completely unrelated or unmatched SLA documents compared with the

service requestor's requirements, it can be skipped; otherwise, it can become a candidate. This means that the relatedness of a provider's SLA produces the service candidate base.

WS-Agreement-compatible SLA documents use XML tags to define service-level contract details. After examining WS-Agreement specifications, we identify two tag types that can be used to define QoS attributes: <wsag:Serviceproperties> and <wsag:GuaranteeTerm>. In other words, QoS-related specifications are likely to be defined using these two tags. From the wording perspective, we consider the QoS definitions associated with the tag <wsag:GuaranteeTerm> have higher assurance over those defined within the tag <wsag:Serviceproperties>. If a QoS attribute is defined within both tags, we will use the value defined within the tag <wsag:GuaranteeTerm>.

The tag *GuaranteeTerm* can be used to state zero or more quantifiable QoS guarantees. In the following example, a service provider of a *FlightReservationService* assures that its response time will be within 2 seconds.

```
<wsag:GuaranteeTerm  Obligated= "wsag:ServiceConsumer">
  <wsag:ServiceScope  ServiceName= "xsd:FlightReservationService">
   …
  <wasg:ResponseTime> "2s" </wasg:QualifyingCondition>
</wsag:GuaranteeTerm>
```

Inside of each of the two tags, individual QoS-related quantifiable attributes can be defined as variables, such as response time and throughout. The tag <wsag:VariableSet> can be used to define a list of (name, value) pairs representing QoS attributes and corresponding values. In the following WS-Agreement definition segment, the <wsag:ServiceProperties> tag defines two QoS attributes: *Reliability* and *ResponseTime*, each being delimited by the tag <wsag:Variable>. As shown in the example, the two QoS variables are grouped into a set represented by the tag <wsag:Variables>.

```
<wsag:ServiceProperties
   wsag:Name= "xs:QoSAttributes"
   wsag:ServiceName= "xs:FlightReservationService">
  <wsag:Variables>
    <wsag:Variable  name= "Reliability"  metric= "job:ReliabilityCount">
      <wsag:Location>
        //TaskDescription/Resources/IndividualReliability/Definition
      </wsag:Location>
    </wsag:Variable>
    <wsag:Variable  name= "ResponseTime"  metric= "job:ResponseTimeCount">
      <wsag:Location>
        //TaskDescription/Resources/IndividualResponseTime/Definition
      </wsag:Location>
    </wsag:Variable>
  </wsag:Variables>
</wsag:WerviceProperties>
```

Therefore, we can define an SLA document as follows.

Definition 1. An SLA document, ws_{sla}, associated with a Web service can be defined as a triple:

$ws_{sla} = (GATT, SATT, ATTV)$, where:

$GATT = \{att_1, att_2, ...att_{N_G}\}$ is a list of guaranteed QoS attributes. $N_G \in I$ is the number of guaranteed attributes defined;

$SATT = \{att_1, att_2, ...att_{N_S}\}$ is a list of specified attributes. N_S is the number of listed attributed defined;

$ATTV = \{< att_1, v_1 >, < att_2, v_2 >, ... < att_{N_G+N_S}, v_{N_G+N_S} >\}$ is a list of (name,value) pairs containing the attribute names and corresponding values specified. The attributes can be either guaranteed attributes or specified attributes.

3.3 SLA Document Parser

Since SLA documents are actually XML files, in general, any XML parser-based tool can be used to fulfill the task. Our previous research creates a Web application code generator WebGen (Zhang and Chung, 2003), which we will adopte it in this work as a SLA document parser.

One big issue of this QoS specification extraction is that the QoS attributes identification is based on variable keywords. For example, a specification of <wsag:Variable name="Reliability"…> defines a QoS attribute *Reliability*. However, for the same meaning, different SLA document writers may use variants of the keyword that may lead to divergences of the same semantic concept. For example, one may choose to use syntactical variants, such as plurals, gerund forms, and past tense suffixes. As an example, different writers may use different forms to define a QoS attribute availability, e.g., "availability," "available," and "availabilities." We chose to partially solve this problem by substituting variable names with their respective stems when we conduct the initial document parsing. A stem is the portion of a word left after the removal of its affixes. Both prefixes and suffixes. For example, "availability" is the stem of "availabilities." Many algorithms have been proposed regarding prefixes and affixes removal. We decided to adopt the Porter stemming algorithm (Porter, 1980) in our research due to its popularity, simplicity, and efficiency. The Porter algorithm, or so-called 'Porter stemmer," removes common morphological and in-flexional endings from English words for term normalization. After the process of word stemming, the tags of an SLA document are changed into a normalized form, with all variants of a word are represented by its stem. In short, word normalization process allows us to capture the semantics of QoS specifications from an SLA document.

We built an SLA document parser to automate the process of QoS attributes elicitation. Our SLA parser takes an SLA document as input, and generates a list of (name, value) pairs comprising the QoS attribute name and corresponding value extracted from SLA document tags. Here we will address how to extract QoS attributes from SLA documents.

Figure 2 summarizes the full parsing and analysis procedure of our SLA document parser. This parser is used to identify two tag types of QoS attributes - <wsag:Serviceproperties> and <wsag:GuaranteeTerm>. The parser first loads an SLA document and parses it into an XML tree. Then it searches the generated tree for the tag <wsag:GuaranteeTerm>. If the tag is found, the sub-tree with the found tag as the root will be examined. The QoS attributes and their corresponding values will be extracted and collected into a list of (name, value) pairs.

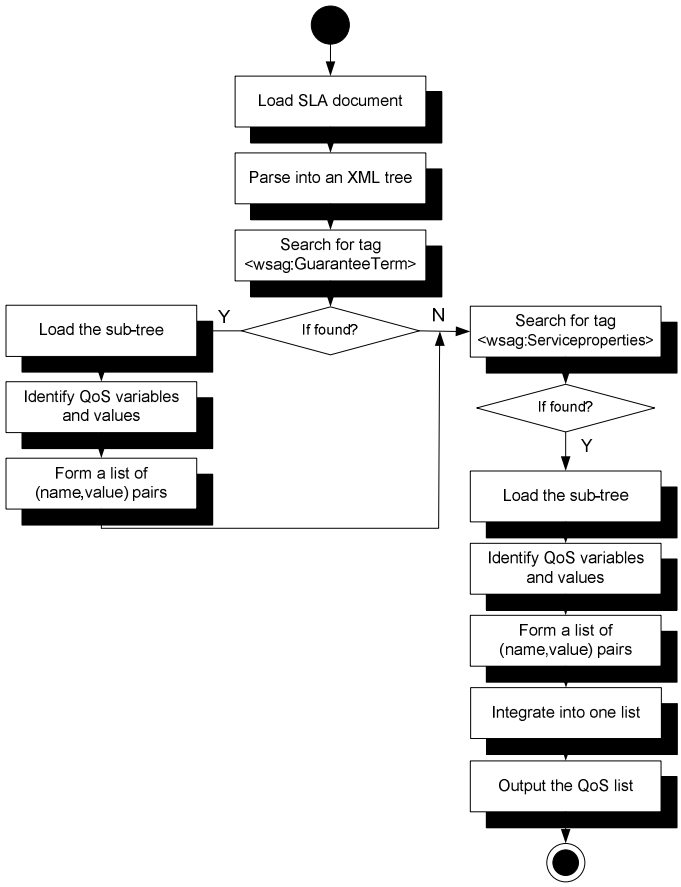

Figure 2. SLA document parsing workflow.

Our parser will proceed to searche the entire XML tree for the other tag <wsag:Serviceproperties>. If the tag is found, the sub-tree with the found tag as the root will also be examined. The QoS attributes and their corresponding values will also be extracted and collected into a list of (name, value) pairs. Then two lists of (name, value) pairs associated with the tweo tags <wsag:Serviceproperties> and <wsag:GuaranteeTerm> will be integrated into one list, with the pair from the first list having higher privileges. Finally, the created list is outputted.

4. QoS-aware Service Discovery and Composition

4.1 Service Discovery

For matching Web services with service consumers' functional requirements, UDDI (Clement et al, 2004) offers consumers 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 the three encoding information, UDDI can support keyword- or directory-based service discovery. However, such service selection process is suitable for text-only service search and it is insufficient to handle query containing numeric computation.

Based on the SLA-based QoS analysis of Web services as presented in previous sections, QoS-aware service discovery can be carried out by two alternatives, namely absolute and relative matchmaking. The absolute matchmaking is a service selection process in which a service is retrieved for a service request if each of the service's QoS attributes fulfills the corresponding requirements of the service request. On the other hand, the relative matchmaking refers to selecting a service for a service request with overall evaluation of the service's QoS attributes. The QoS-aware service discovery with absolute matchmaking can be represented as follows.

$Match_{num}(Match_{txt}(Match_{txt}(Sr, R1), R2), R3)$
$Match_{txt}$: *S x R→S, UDDI-based matchmaking*
$Match_{num}$: *S x R→S, arithmetic-based matchmaking*
S: a set of services
R: a set of requirements
Sr: the set of services in a service registry
R1: functional requirements of a service request
R2: text-based QoS requirements of a service request
R3: numeric based QoS requirements of a service request

A service request is separated into three parts. R1 is the set of functional requirements; R2 is the set of QoS requirements encoded by text-based data types, including authentication, confidentiality, integrity, and non-repudiation; R3 is the set of numeric-based QoS requirements, including response time, throughput, reliability, availability, and price. For text-based requirements (i.e., R1 and R2), keyword or directory-like service discovery from UDDI (i.e., $Match_{txt}$) is employed to filter out undesired services and a set of candidate services will be available after the process. For numeric-based requirements (i.e. R3), the basic arithmetic (i.e., $Match_{num}$) is applied to determine whether a service fulfills a service request as shown below.

A service s is selected by $Match_{num,}$ *if the following inequalities are true for each numeric QoS attribute q in s and the corresponding requirement r in R3.*
q.value >= r.value for positive QoS attributes
q.value <= r.value for negative QoS attributes

The positive QoS attributes (e.g., throughput, reliability and availability) indicate that the higher the attribute value, the better the quality. In contrast, the negative QoS attributes (e.g., response time and price) indicate that the higher the attribute value, the worse the quality.

The relative matchmaking provides a more flexible service selection method for text-based requirements. The relative matchmaking is defined as follows.

$Match_{num\_mcdm}(Match_{txt\_rough}(Match_{txt\_rough}(Sr, R1), R2), R3)$
$Match_{txt\_rough}$: *S x R→S, enhanced UDDI matchmaking*
$Match_{num\_mcdm}$: *S x R→S, MCDM based matchmaking*
S: a set of services

R: a set of requirements
Sr: the set of services in a service registry
R1: functional requirements of a service request
R2: text based QoS requirements of a service request
R3: numeric based QoS requirements of a service request

In addition to keyword or directory-like service discovery, the enhanced UDDI matchmaking (i.e., $Match_{txt\_rough}$) allows requirements with wild characters to carry out partial match in the service selection process. Wild characters can be inserted in any place of functional and text-based QoS requirements to express more general queries. For numeric-based requirements, a multiple-criteria decision making (MCDM) technique with weighted sum model (WSM) is employed to perform an overall evaluation within two steps as shown below (Hwang and Yoon, 1981).

Step 1: Normalization of each QoS attribute in a candidate service

The value of each numeric QoS attribute, *q*, in 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 QoS 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.

Step 2: Weighting and sum of each QoS attribute in a candidate service

Each normalized numeric QoS attribute, *q*, in a candidate service, *s*, multiplies the corresponding weight, *w*, given by a service consumer will generate an overall evaluation score of the service as shown below.

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

$Match_{num\_mcdm}$ will select services whose evaluation scores are greater than the threshold score given by the service consumer.

The current UDDI standard has offered partial match functionality with wild characters, but it 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.

4.2 Service Composition

Service composition is a process of creating new functionalities by aggregating several independent services. In the process, various workflow patterns are applied to configure these services into a new composite service with value-added functionalities. From a service consumer's perspective, the QoS performance of a composite service is

perceived aggregately from the performance of its constituent services. Thus, service selection for a QoS-aware service composition should be carried out with a global view of QoS attributes (Menasce, 2004). In general, the service selection depends on numeric-based attributes only, there is no aggregative effect of text-based attributes for QoS-aware service composition. For example, the performance of two interrelated services' authentication capability is always perceived consistently regardless of their composing patterns. To overcome the shortage of non-aggregative composition, we utilize our Petri nets-based composition patterns (Yang, 2006), and based upon the composition patterns, we present calculation of aggregative effects of QoS attributes accordingly.

Figure 3 summarizes eight workflow composition patterns using Petri nets representations (Yang, 2006): (1) sequence, (2) split, (3) split-join, (4) unordered, (5) choice, (6) if-then-else, (7) iterate, and (8) repeat until. Based on the formal semantics of these patterns, we can derive the corresponding aggregative effect of numeric QoS attributes as shown in Table 2.

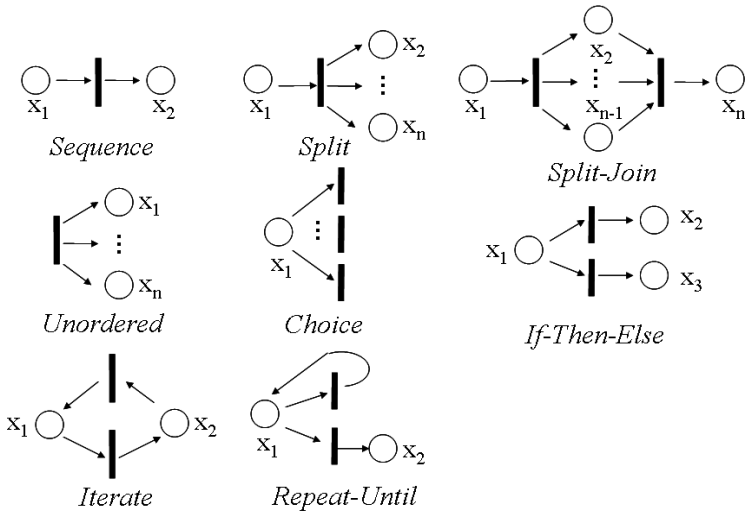

Figure 3. Workflow patterns with Petri nets

Table 2. Aggregative effect of numeric QoS attributes

Attributes / Patterns	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	$n * x_1 + x_2$	$\min\{x_1, x_2\}$	$x_1^n * x_2$	$x_1^n * x_2$	$n * x_1 + x_2$

A lot of services with identical functionality may be available for a task so service consumers usually has numerous choices between different sets of services in a service composition. By Table 2, customers will be able to estimate the QoS performance of different sets of services and apply the absolute or relative service selection methods as specified in Section 3 to determine which set of services is satisfied with their requirements.

Jaeger, Rojec-Goldmann and Muhl (2004) also identify the aggregation of numerical QoS dimensions for some workflow patterns, but their aggregation is not based upon a consensus of workflow pattern definitions (Staab, 2003). Jaeger, Rojec-Goldmann and Muhl (2005) proposed a more precise QoS aggregation method by considering dependencies between services. They claimed that the QoS performance of two services such as uptime probability should not be estimated aggregately if they are located in same server. However, service consumers do not need to consider the information of dependencies because the details of a Web service is supposed to be a black-box in standard Web services model. Zeng et. al. (2004) discussed the computational complexity problem of choosing the best set of services by proposing an integer programming based solution to select an optimal execution plan with lower complexity. They pointed out that the volume of sets of services for a service composition is proportional to the amount of available services for the corresponding tasks, thus the computational complexity of a brute-force estimation method is exponential.

A QoS-aware service composition solution should define formal semantics of different workflow patterns and provide the corresponding aggregative effects as well. Besides, how to assist service consumer in selecting qualified sets of services for a service composition with low computational complexity should also be studied further.

5. Conclusions

The development of QoS-aware Web services is a popular research as it is seen as the foundation toward trustworthy Web services. The promise of providing services with certain QoS performance will make service consumers be more confident when they adopt Web services for critical tasks. In order to benefit both service providers and service consumers, a general QoS model of Web services is required. The model should balance different viewpoints from the two parties and provide formal definitions of QoS attributes 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 QoS 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 further failure recovery will be worth studying.

References

Cardoso, J., Miller, J., Sheth, A. and Arnold, J. (2002) 'Modeling Quality of Service for Workflows and Web Service Processes', *LSDIS lab, Computer Science, University of Georgia*, Tech. Rep. #02-002.

Chinnici, R. et al. (ed.) (2006) 'Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language', *WWW Consortium*, March, http://www.w3.org/TR/wsdl20/

Clement et al. L. (2004) 'UDDI Version 3.0.2', *OASIS*, October, http://uddi.org/pubs /uddi_v3.htm

Garvin, D. A. (1988) *Managing quality: The strategic and competitive edge*, Free Press, New York.

GGF. (2005) 'Web Services Agreement Specification (WS-Agreement)', *Global Grid Forum*. http://www.ggf.org.

Hwang, C.L. and Yoon, K. (1981) *Multiple attribute decision making: Methods and applications*, Springer-Verlag.

IBM. (2003) 'Web Service Level Agreement (WSLA) Language Specification', http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

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

Jaeger, M.C., Rojec-Goldmann, G. and Muhl, G. (2005) 'QoS aggregation in Web service compositions' *Proceedings of IEEE International conference on e-Technology, e-Commerce and e-Service (EEE)*, pp. 181-185.

Jay, F. and Mayer, R. (1990) 'IEEE Standard Glossary of Software Engineering Terminology', IEEE Std 610.12-1990.

Menasce, D. A. (2002) 'QoS Issues in Web Services', *IEEE Internet Computing*, Vol. 6, Issue 6, pp. 72-75.

Menasce, D. A. (2004) 'Composing Web Services: A QoS View', *IEEE Internet Computing*, Vol. 8, Issue 6, pp. 88-90.

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

Pezzini, M. (2003) 'Composite Applications Head Toward the Mainstream', *Gartner, Inc*. October, http://www.gartner.com

Porter, M. (1980) 'An Algorithm for Suffix Stripping Program', *Automated Library and Information Systems*. 14(3): 130-137.

Staab, S. (ed.) (2003) 'Web Services: Been There, Done That?', *IEEE Intelligent Systems*, Vol. 18, Issue 1, pp. 72-85.

Wikipedia. (2006) 'Quality of Service', August, http://en.wikipedia.org/wiki/Quality of_service

WS-Agreement. (2005) 'Web Services Agreement Specification (WS-Agreement)', http://www.ggf.org/Public_Comment_Docs/Documents/Oct-2005/WS-AgreementSpecificationDraft050920.pdf.

Yang, S.J.H., Hsieh, J.S.F., Lan, B.C.W. and Chung, J.Y. (2006) 'Composition and Evaluation of Trustworthy Web Services', *International Journal of Web and Grid Services*. Vol. 2, No. 1, pp. 5-24.

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

Zhang, J. (2005) 'Trustworthy Web Services: Actions for Now', *IEEE IT Professional*, Vol. 7, Issue 1, pp. 32-36.

Zhang, J. and Chung, J.-Y. (2003) 'Mockup-driven Fast-prototyping Methodology for Web Application Development', *Software Practice & Experience Journal*. 33(13):

Stephen J.H. Yang, Jia Zhang, Blue C.W. Lan

1251-1272.