2005

# Dealing with Web service QoS factors using constraint hierarchy

Y. Guan
*University of Wollongong*, yguan@uow.edu.au

Aditya K. Ghose
*University of Wollongong*, aditya@uow.edu.au

# Dealing with Web service QoS factors using constraint hierarchy


## Abstract

Functionality and non-functional properties are two critical factors in web service technology, but non-functional properties (quality factors) are often ignored. Usually, these are articulated as statements of objectives, as opposed to prepositional assertions. A key challenge in dealing with objectives is that there is no obvious means to decide when they are satisfied. In effect, these objectives are never fu lly satisfied, but satisficed to varying degrees. Alternative design decisions need to trade-off varying degrees of satisfaction of potentially mutually contradictory non-functional requirements. In some circumstances, non-Junctional properties are crucial; they do affect the design decision. Upon a request, there are a range o f web services that might provide the required functionality, so the web service selection can only be done based on their Quality of Service (QoS). Therefore, a quality-based web service model is in high demand. The key contribution of this paper is the use of the hierarchical constraint logic programming framework [9, 10] in dealing with quality factors. We show how quality factors can be formulated as soft constraints and how the machinery associated with constraint hierarchies can be used to evaluate the web services.



## Disciplines

Physical Sciences and Mathematics

## Publication Details

Guan, Y. & Ghose, A. K. (2005). Dealing with Web service QoS factors using constraint hierarchy. In W. Wong, W. Chu & N. Juristo (Eds.), International Conference on Software Engineering and Knowledge Engineering (pp. 578-583). USA: Knowledge Systems Institute Graduate School.

# Dealing with Web Service QoS factors using Constraint Hierarchy

Ying Guan, Aditya K. Ghose
Decision Systems Laboratory
School of IT and Computer Science
University of Wollongong, Australia
{yg32, aditya}@uow.edu.au

## Abstract

*Functionality and non-functional properties are two critical factors in web service technology, but non-functional properties (quality factors) are often ignored. Usually, these are articulated as statements of objectives, as opposed to prepositional assertions. A key challenge in dealing with objectives is that there is no obvious means to decide when they are satisfied. In effect, these objectives are never fully satisfied, but satisficed to varying degrees. Alternative design decisions need to trade-off varying degrees of satisfaction of potentially mutually contradictory non-functional requirements. In some circumstances, non-functional properties are crucial; they do affect the design decision. Upon a request, there are a range of web services that might provide the required functionality, so the web service selection can only be done based on their Quality of Service (QoS). Therefore, a quality-based web service model is in high demand. The key contribution of this paper is the use of the hierarchical constraint logic programming framework [9, 10] in dealing with quality factors. We show how quality factors can be formulated as soft constraints and how the machinery associated with constraint hierarchies can be used to evaluate the web services.*

## 1 Introduction

Web services are self-describing software applications that can be advertised, located, and used across the Internet using a set of standards such as SOAP, WSDL [13], and UDDI [16]. Web services are built in a distributed environment of Internet, including sets of platform independent software components. Web services are regarded as a fairly new and promising technology. They have many prominent advantages over traditional World Wide Web services. However, although they have generated a lot of interest and are becoming increasingly popular, there are some factors that affect their adoption rate mentioned in [15]. One of the issues is the quality of web services. Quality of service (QoS) [4] is a combination of several qualities or properties of a service: such as capability, performance, reliability, integrity, security, and so on. At the present time, UDDI based look ups for web services are based on the functional aspects of the desired Web services without caring about the quality of the service. But quality of a service is extremely important. Upon the request of a web service, there are a range of web services that might provide the required functionality. Under these circumstances, the web service selection can only be done based on their Quality of Service (QoS). Therefore, a quality-based web service model is in high demand.

Quality factors of web services can be specified as the quality constraints about the functionality of those web services, and some of those constraints can be quantitatively expressed. Given requirements of a web service, functional ones and non-functional ones, as far as non-functional requirements are concerned, the user usually states not only the quality demanded but also their preference. Therefore, quality factors and the preference level cannot be considered separately.

The key contribution of this paper is the use of the hierarchical constraint logic programming framework [9, 10] in dealing with quality factors. Constraint logic programming was developed to extend the ability of traditional logic programming to deal with knowledge (facts and rules) expressed as Horn clauses with specially designated *constraint predicates*. The resulting systems were more efficient than standard logic programming systems because of their ability to use special-purpose *constraint solvers*, which, in effect, understood the "meaning" of the constraint predicates, and dealt with them in more efficient ways than the resolution proof procedure that most logic programming systems relied on. Constraint logic programming also offered better expressivity. Hierarchical constraint logic programming (HCLP) was developed to deal with the fact that many of the constraints articulated by users in real-life problems are *soft constraints*, i.e., these were constraints that one would ideally seek to satisfy, but which could be violated (or satisfied to a lesser degree) if absolutely necessary. Soft constraints typically have varying degrees of priority, hence the HCLP framework permits the specification of *constraint hierarchies*, i.e., sets of soft constraints labelled with varying degrees of priority. This property can be used to express the preference of web service requestor efficiently. Our larger project seeks to deploy the full capability of the HCLP framework in dealing with quality factors. In the current paper, for the sake of brevity, we only focus on the constraint hierarchy component of framework. Our focus is on showing how quality factors can be formulated as soft constraints and how the machinery associated with constraint hierarchies can be used to evaluate the web services in our proposed model. Also, we apply this model to web service selection and web

service composition to illustrate how this model works on these two applications.

This paper is organized as follows. In Section 2, we give a brief introduction to QoS and constraint hierarchy and also give the details of our model QoSCH. In Section 3, we apply our model for selection and branch and bound composition of web service. Finally, we discuss related work in Section 4 and conclude in Section 5.

## 2 Quality of Services (QoS) and Constraint Hierarchy (CH)

Different web services have different demands for QoS and the priorities of each quality factor are also changeable. Therefore, a quality-based web service model that satisfies these two aspects is needed. Our proposed model QoSCH integrates the hierarchical constraint hierarchies into QoS attributes. In the following three subsections, we give a brief review of QoS and constraint hierarchies and then details of the QoSCH model.

**Table 1**. Possible Measures for Quality attributes

| Quality attribute | Possible Measures |
|---|---|
| Security | probability of detecting attack, percentage of services available under denial-of services attack; extent to which services damaged and/or legitimate access denied |
| Performance | responsetime,latency,throughput, execution duration/time |
| Availability | time interval when the system must be available, available time, time interval in which system can be in degraded mode, repair time, task time, number of problems solved |
| Cost | cost in terms of elements affected,effort,money; execution price |
| Accuracy | number of error ,rate of fail or successful operations to total operation, amount of time/data lost |
| Usability | use system efficiently, minimize impact of errors |
| Reliability | Mean time between failure, Mean time to failure, Mean time to Transaction |

### 2.1 QoS

Requirements consist of functional requirements and non-functional requirements [3] (quality factors). NFR can be specified as a constraint expressed on the functional requirements of a system. As all requirements can be specified in measurable terms, but for NFRs, there are no measurable ways to determine when it is met. No solution can be said to optimally achieve certain NFR. A key challenge in dealing with NFRs is articulating them in terms of metrics, on which one could then apply thresholds or seek to maximize or minimize. In Table 1, we list possible measures for some NFRs, along the lines of the proposal in [1]. Those possible metrics would permit us to formulate constraint-style representations of NFRs. In this table, we only list part of those attributes that are easy to be specified using numbers, while there are still other attributes that are difficult to be expressed in explicit numeric ways, for instance, confidentiality, portability, etc. We believe that quantitative metrics for these can also be developed in the future, adding strength to our proposal.

### 2.2 Constraint Hierarchy

Constraint hierarchies (CHs) belong to traditional frameworks for the handling of over-constrained problems. They allow us to express hard constraints which have to be satisfied and several preference levels of soft constraints which violations are minimized level by level subsequently [5]. To introduce the constraint hierarchies, we will use the definition of constraint hierarchies in [10].

A *constraint hierarchy* is a finite set of labeled constraints. A *labeled constraint* is a constraint labeled with a strength, written $lc$ where $c$ is a constraint and $l$ is a strength. In this paper, we use the definition of strength and constraint with a strength of [6] which uses an integer $k$ ($0 \le k \le l$, $l$ is a constraint positive integer) as strength of a constraint instead of using symbols such as required, strong, medium and weak as the strength as in [10]. Given a constraint hierarchy H, $H_0$ is a vector of required constraints in H, in some arbitrary order, with their labels removed. Similarly, $H_1$ is a vector of strongest non-required constraints in H up to the weakest level $H_n$, where n is a number of non-required levels in the hierarchy H. A valuation for a set of constraints is a function that maps free variables in the constraints to elements in domain $D$ over which the constraints are defined. A solution to a constraint hierarchy is such a set of valuations for the free variables in the hierarchy that any valuation in the solution set satisfies at least the required constraints. An error function $e(c\theta)$ is used to indicate how nearly constraint c is satisfied for a valuation $\theta$. This function returns a non-negative real number and must have the property that $e(c\theta) = 0$ if and only if $c$ holds. Major error functions are the predicate and metric error. In our model, we adopt the metric error function. The metric function is mainly adopted for arithmetic constrains composed of arithmetic functions and relations [6]. It expresses constraint errors as some distances. Typically, for arithmetic equality constraints, it uses the differences between the left- and right- hand sides. For example, the error of the constraint x = y may be given as follows: $e("x=y", \theta) = |\theta(x) - \theta(y)|$. When the domain $D$ is a metric space with distance function dist, metric error function may be defined. A normalization of domain $D$ has to be done to obtain metric space with suitable distance function.

CHs define the so called comparators aimed to select solutions (the best assignment of values to particular variables) via minimizing errors of violated constraints. If a solution θ is better than a solution σ, there is some level k in the hierarchy such that for $1 \le i<k$, $g(E(H_i\theta)) <_g g(E(H_i\theta))$, and at level k, $g(E(H_k\sigma)) <_g g(E(H_k\sigma))$. Currently, there are three groups of comparators: global, local and regional comparators. For a local comparator, each constraint is considered individually, for a global com-

parator, the errors for all constraints at a given level are aggregated using the combining function g. For a regional comparator, each constraint at a given level is considered individually. There are a number of comparators defined by combining function g and the relations <>g and <g (the symbol <> means equal). The global comparator includes weighted-sum-better (WSB), worse-case-better (WCB) and least-squares-better (LSB). Due to space limitation, we omit the formal definitions of these comparators. In our model, we use the global metric comparator, which aggregate errors of violated constraints at each level.

### 2.3 QoSCH Model

Constraint hierarchy can be well used to express the quality factors constraints in a user defined preference level. And those constraints of each hierarchy level are not fixed. They can vary with the non-functional requirements of different web services. QoSCH is mainly used to construct the requirements of each web service using constraint hierarchy. It consists of two parts. One is the functional requirements part, which states the functionality of the web service, for example, online booking, view transaction history, etc. The other part is quality part, which states the quality factors of this web service, such as security, accuracy and so on. As mentioned in Section 2.1 that not all variables relating to QoS factors can be assigned values as mentioned in QoS section, so we will use a projection on the whole set of QoS factors to construct the constraint hierarchy. In Table 1, we have listed some possible measures for those quality factors that are easy to be measured quantitatively. Each web service, requested web services or provided ones, can all be expressed using this model.

Those values of each constraint hierarchy will be provided in different ways. The main means for gathering them might be a requester defining the requested web service and providers providing, user feedback and detection by certain monitor for competitive web services. Before constructing QoSCH models for each web services, the requester and providers should offer those values for their constraint hierarchies.

Using QoSCH model, we can measure the performance of a web service by calculating the constraint hierarchy distance from the requested web service. This measurement process is defined as following:

1. Construct the QoSCH models for both anticipant web service and available service.
2. If the functionality of the available web service meets all the functionality of the anticipant web service, then do the following steps, otherwise, this available web service is not qualified.
3. Calculate the distance from the constraint hierarchy of the available web service to the constraint hierarchy of the anticipant web service.

The calculation of distance applies the comparators of constraint hierarchy stated in the previous section. To do this, firstly, the available service should be characterized with the same set of non-functional properties with the valuation $\sigma$ for the variables in the constraint hierarchy; secondly, calculate the distance of the value $\sigma$ from the constraint hierarchy to evaluate how far its quality performance is from the proposed web service. In our model, we use an error function to define the error rate of valuation $\sigma$ to proposed valuation $\theta$, $e(\theta(x)=\sigma(x))=|\theta(x)-\sigma(x)|/\theta(x)$.

For example, given a constraint hierarchy, HC = {x ≤ 2 strength 1; y ≤ 0.02 strength 0.8; z ≥ 20 strength 0.5} and a value $\sigma$ = {x = 1, y = 0.025, z = 18}, the distance of $\sigma$ from HC is:

$d=1\times 0+0.8\times|0.02-0.025|/0.02+0.5\times|20-18|/20=0.25$

Note that, if a constraint at the required level (strength 1) is not held, then this value should be discarded, the distance of it is meaningless.

However, there still exists another case, that is, some of the web services meet the functionality, but they do not have a value for some of the constraints. For these kinds of web services, they are still qualified for competition as long as the constraint they miss is not in the required level. Moreover, the error rate for the missing constraint will be set as the highest error rate, 1. Similarly, if a web service provides more constraints than the requester required, then this will be considered as extra criteria for selection when with the current constraints, the error distance is not sufficient for selection, that is, there is more than one web service that holds the same smallest distance.

## 3 Applications for Constraint Hierarchy in QoS

Using the QoSCH model will make web services more efficient in many ways, for example service selection and dynamic service composition.

### 3.1 Web Services Selection

Given the functional requirements for a requested web service, there may be many available services that can provide the expected functionality. The only difference among them is the quality factors of the service. Using the model we have proposed, we can select a relatively optimal service from the set of available services. The selection steps are:

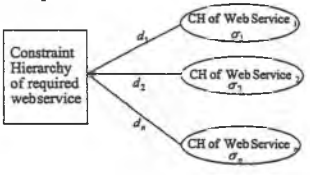


Fig.1. Web services selection

1. Construct the QoSCH model for each web service, requested one and available ones.
2. Select those services that meet all the functional requirements. Calculate the distances from the constraint hierarchy of each web services selected on

step 1, select the web service with minimum distance ($\min(d_i)$)(Figure 1).

Here we use an example to illustrate how this selection process works. In this example, the anticipant web service is an online payment service. This web service is supposed to provide online credit card payments and relevant services, such as inquiry about accounts, view transactions history, etc. From its functional requirements, we can see that the service needs high quality on performance, security, accuracy and so on. Therefore, the quality constraint for this service is high performance, low cost, low transaction fee, high accuracy, high security, robustness, good reputation, high availability and so on. The constraint hierarchy on Table 2 is the projection of variables that can be specified with quantifiable parameters.

**Table 2.** QoSCH model for Online Payment Service

| Functional Requirements | Constraint Hierarchy of Quality Factors | |
|---|---|---|
| Accept credit cards | Constraints | Strength |
| Process credit cards | response time $\leq$ 0.1s | 1 |
| Inquire about account | probability of detecting attack=1 | 0.9 |
| Payment execution | error rate $\leq$ 0.001% | 0.9 |
| View transaction history | execution prize transaction amount × 0.5% | 0.8 |
| | concurrent transaction $\geq$ 10000 | 0.7 |
| | bandwidth $\geq$ 56kbyte | 0.6 |
| | cost $\leq$ $200/month | 0.5 |

**Table 3.** Valuations for constraints variables of $OPS_1$, $OPS_2$ and $OPS_3$

| $OPS_1$ | $OPS_2$ | $OPS_3$ |
|---|---|---|
| response time = 0.1s | response time = 0.1s | response time = 0.2s |
| probability of detecting attack = 0.98 | probability of detecting attack = 0.99 | probability of detecting attack = 0.99 |
| error rate < 0.002% | error rate < 0.0015% | error rate < 0.0016% |
| execution prize =transaction amount × 1% | execution prize = transaction amount × 1% | execution prize = transaction amount × 1% |
| - | concurrent transaction =8000 | concurrent transaction =9000 |
| bandwidth =300kbyte | bandwidth = 80kbyte | bandwidth = 256kbyte |
| cost = $220/month | cost = $240/month | cost = $200/month |

With the functional requirements and constraint hierarchy of the quality of this online payment service, we can begin to do the selection from a set of available services. After the first step of service selection, there are three services left, $OPS_1$, $OPS_2$ and $OPS_3$. All of them meet the functional requirements of the online payment service. Then we need to compare the distance of the constraint hierarchy of them. The constraint hierarchies for these two services are listed in Table 3. In the required (strength 1) level, *response time* of $OPS_3$ does not satisfy this constraint, therefore, $OPS_3$ is eliminated through selection. While the other two web services satisfy constraints in the required level. $OPS_1$ fails to provide the value for *constraint concurrent transaction*, therefore, the error rate of this item is 1. After comparing with the constraint hierarchy of anticipant web service and these two available services using the global metric comparator, we can get the two distances $d_1$ and $d_2$ for $OPS_1$ and $OPS_2$ respectively. $d_1$=1.42, $d_2$=1.109. $d_2 < d_1$, So $OPS_2$ is selected.

### 3.2 Web Services Composition

Web service composition is the ability of one business to provide value-added services through composition of basic web services, possibly offered by different companies [12]. A composite web service is an aggregation of web services which interact with each other based on a process model [7]. Web service standards, such as UDDI, WSDL, SOAP, do not deal with the composition of existing services. Many researchers have worked on this problem ([2], [7], [8]).

In this section, we propose Branch and Bound Web Services Composition (BBWSC) by using QoSCH model for web service composition. Quality constraints and preferences are assigned to composite services rather than to individual tasks within a composite service. [7]

The branch and bound [14] composition process consists of two steps:

1. Find the first composite service that meets all FRs and hard constraints. Let the distance from constraint hierarch be $d_i$.
2. Try to construct another composite service for the same requirements. At each step, compare distance d with $d_i$, if $d > d_i$, then prune this branch.

The formal definition for BBWSC is shown as follows.

**Definition**: Let S be a partially composed service, i.e., not all variables relating to QoS factors have been assigned values. Let Var(S) denote the set of variable which have been assigned values in S. Let Project (H, V) be the projection of constraint hierarchy H on the set of variables V. Use the following algorithm to do the web service composition.

```
S := null service;
d := ∞;
while alternative service composition exist do
  S := φ
  while ¬complete (S) do
    S := S ⊕ s
      [execute a service composition step by
       combining using operator ⊕ S with s]
    if distance(Project(H,Var(S)),S) > d then exit
  d := distance(Project(H,Var(S)),S)
return S
```

When executing the web services composition, we adopt the aggregation functions proposed in [7] for each step composition.

Now we use an online shopping service as an instance to utilize our web service composition. This Online Shopping Service (OSS) is to provide shopping online service for an electronic shopping website. This website needs to provide customers with a free online shopping space, so that customers can search for their desired products and purchase them. Figure 2 shows the elementary process of this online shopping service. OSS needs three successive independent web services, Register/Login/Logout Service (RLS), View Product Service (VPS) and Online Payment Service (OPS). VPS is made up of two web services, Search Engine Service (SES) and Products Distributor Service (PDS).

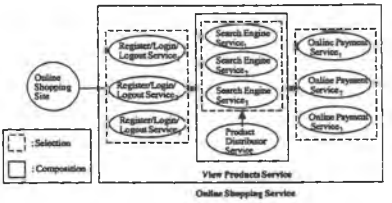


Fig.2. Requirements of Online Shopping Service

The quality factors of OSS that stakeholders care about are: response time, probability of detecting attack, error rate, execution prize, concurrent transaction, bandwidth, cost, reputation, integrity, capability, robustness. These factors are the set of variables V as mentioned in the algorithm. So we need to get the projection of those factors that have been assigned on V. The result of the projection is listed in Table 4. Similarly, we can get the projection of constraint hierarchy for VPS (Table 8), and the constraint hierarchies for each service (Table 5, Table 6, Table 7 and Table 9).

**Table 4.** QoSCH Model of Online Shopping Services

| **Functional Requirements** | **Constraint Hierarchy of the Quality of Service** | |
|---|---|---|
| Register | **Constraints** | **Strength** |
| Payment | response time ≤ 0.2s | 1 |
| View history | probability of detecting attack = 1 | 0.9 |
| Login/Logout | error rate ≤ 0.001% | 0.9 |
| Inquire account | execution prize ≤ transaction amount × 0.5% | 0.8 |
| Search products | concurrent transaction ≥ 10000 | 0.7 |
| Accept credit cards | bandwidth ≥ 256kbyte | 0.6 |
| Process credit cards | cost ≤ $500 | 0.5 |

**Table 5.** Valuations for constraints variables of $RLS_1$ and $RLS_2$

| $RLS_1$ | $RLS_2$ |
|---|---|
| response time = 0.1s | response time = 0.1s |
| probability of detecting attack = 0.99 | probability of detecting attack = 0.95 |
| error rate ≤ 0.001% | error rate ≤ 0.002% |
| concurrent transaction = 12000 | concurrent transaction = 10000 |
| bandwidth = 512kbyte | bandwidth = 360kbyte |
| cost = $50/month | cost = $45/month |

**Table 6.** Valuations for constraints variables of $SES_1$ and $SES_2$

| $SES_1$ | $SES_2$ |
|---|---|
| response time = 0.1s | response time = 0.1s |
| error rate ≤ 0.01% | error rate ≤ 0.009% |
| concurrent transaction= 8000 | concurrent transaction= 9000 |
| bandwidth = 80kbyte | bandwidth= 300kbyte |
| cost = $150/month | cost = $120/month |

**Table 7.** Valuations for constraints variables of $SES_3$ and PDS

| $SES_3$ | PDS |
|---|---|
| response time = 0.2s | response time = 0.1s |
| error rate ≤ 0.012% | probability of detecting attack = 0.9 |
| concurrent transaction = 8500 | error rate ≤ 0.002% |
| bandwidth = 320kbyte | concurrent transaction= 10000 |
| cost = $100/month | bandwidth = 512kbyte |
| | cost = $150/month |

**Table 8.** QoSCH Model of View Product Service

| **Functional Requirements** | **Constraint Hierarchy of VPS** | **Strength** |
|---|---|---|
| Search | response time ≤ 0.2 | 1 |
| Display search items | probability of detecting attack= 1 | 0.9 |
| Distribute product | error rate ≤ 0.001% | 0.9 |
| | concurrent transaction ≥ 10000 | 0.7 |
| | bandwidth ≥ 256kbyte | 0.6 |
| | cost ≤ $200/month | 0.5 |

**Table 9.** Valuations for constraints variables of composite web services $SES_1$+PDS, $SES_2$+PDS and $SES_3$+PDS

| $SES_1$+PDS | $SES_2$+PDS | $SES_3$+PDS |
|---|---|---|
| response time =0.2s | response time =0.2s | response time =0.3s |
| probability of detecting attack = 0.99 | probability of detecting attack = 0.99 | probability of detecting attack = 0.99 |
| error rate ≤ 0.003% | error rate ≤ 0.002% | error rate ≤ 0.004% |
| concurrent transaction = 8500 | concurrent transaction = 9000 | concurrent transaction = 8500 |
| bandwidth = 280kbyte | bandwidth = 300kbyte | bandwidth = 320kbyte |
| cost = $300/month | cost = $270/month | cost = $250/month |
| d = 2.164 | d = 1.154 | - |

In this example, there are two anticipant composite services, view products service and online shopping service. Firstly, let us look at the simple one, view product service. According to the definition defined above, the

first thing we need to do is to find out composite services that meet all the requirements that the required services needs. There are three composite services that satisfy this requirements, they are, Search Engine Service$_1$ and Product Distributor Service Engine Service$_2$ and Product Distributor Service, Engine Service$_3$ and Product Distributor Service. The distances from constraint hierarchy of the first two services are 2.164 and 1.154 respectively, while the third one is discarded, because it does not satisfy the required constraint *response time*. So Engine Service 2 and Product Distributor Service are the final composite services for View Products Service.

There is only one step in VPS composition, so we cannot see the branch and bound process clearly, now, let us look at the OSS composition. Firstly, we find the first suitable composite web service RLS$_1$+VPS+OPS$_1$, the distance for it from the required service constraint hierarchy is 2.958. Then we find another alternative composite service RLS$_1$+VPS+OPS$_2$ with the distance 2.409. This service is selected as the optimal composite services temperately. The composite services RLS$_2$+VPS+OPS$_1$ also meet all the requirements. But when we check the second step of composition, the distance is 2.58(> 2.39), so it is prune. Another alternative composite service RLS$_2$+ VPS+OPS$_2$ is the same case. So these two composite services are pruned before the whole composite process finishes.

## 4 Related Work

Functionality and non-functional properties are two essential factors for web services. Functionality is used to measure whether a web service meets all the functional requirements of an anticipant web service, while non-functional properties are used to evaluate the performance of the web service. This has been viewed as an efficient means to distinguish functional similar web services. Quality-driven web service selection and composition have received considerable recent attention. Much work has been done to take QoS factors into consideration for selection and composition of web service. In [15], the author proposed a QoS model which offers a QoS certifier to verify QoS claims from the web service suppliers. This approach lacks the ability to meet the dynamics of a market place where the need of both consumers and providers are constantly changing [17]. In [7], authors proposed a global planning approach to optimally select component services during the execution of a composite service. The proposed approach is quality-driven and using Multiple Attribute Decision Making (MADM) [11] approach to select optimal execution plan. But it is not very efficient for large scale composite services, because it requires generating all possible execution plans, the computation cost is high. Whereas, our BBWSC, branch and bound [14] based web services composition, improve the composition efficiency in great extent.

## 5 Conclusions

The quality factors of web services have become increasingly important. There are two main aspects about them that should be taken into account. One is the quality factors themselves; the other is the preference extent from the web service requester's view. In this paper we proposed a model QoSCH which uses a constraint hierarchy to specify the quality factors of a web service. The constraint hierarchy fulfills the task of achieving these two aspects well. We also applied QoSCH to web services selection and web services composition. For web services composition, we integrate branch and bound search with the constraint hierarchy model to do this job. In this paper, we only focused on those quality factors that are easy to be specified quantitatively. There are still many other quality factors, such as, reliability, portability, capability, etc., which are not mentioned in this paper. These are the subjects of future work.

## References


[1] Bass Len, Paul Clements, Rick Kazman, *Software architecture in practice*, Boston : Addison-Wesley, c2003

[2] Biplav Srivastava, Jana Koehler, *Web Service Composition Current Solutions and Open Problems*, ICAPS 2003

[3] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, 2000.

[4] Daniel A. Menasce, *QoS Issues in Web Services*, IEEE Internet Computing, and November. December 2002

[5] Hana Rudová, *Constraint Satisfaction with Preferences*,Ph.D. Thesis,2001

[6] Hiroshi Hosobe, *A foundation of Solution Methods for Constraint Hierarchies*, Kluwer Academic Publishers 2003

[7] Liangzhao Zeng, Boualem Benatallah, Marion Dumas, Jayant Kalagnanam, Quan Z. Sheng, *Quality Driven Web Services Composition*, International World Wide Web Conference archive,Proceedings of the twelfth international conference on World Wide Web ,411 - 421 , 2003

[8] Michael C. Jaeger, Gregor Rojec-Goldmann, Gero Mühl, *QoS Aggregation for Web Service Composition using Workflow Patterns*, Enterprise Distributed Object Computing Conference, Eighth IEEE International (EDOC'04) , 2004

[9] Molly Wilson, Alan Boring, *Hierarchical Constraint Logical Programming*, Journal of logic programming,1993

[10] Molly Wilson, *Hierarchical Constraint Logic Programming*, Technical Report 93-05-01, University of Washington, May 1993. (PhD Dissertation).

[11] M. Köksalan and S. Zionts, editors. *Multiple Criteria Decision Making in the New Millennium* .Springer-Verlag, 2001

[12] Paulo F. Pires, Mario R.F. Benevides, Marta Mattoso, *Building Reliable Web Services*, Web, Web Services and Database Systems 2002

[13] R.Chinnici et al., *Web Service Description Language (WSDL) Version 1.2*, World Wide Web Consortium, 2002, www.w3.org/TR/wsdl12/.

[14] Russell, Stuart J., *Artificial intelligence : a modern approach*, Prentice Hall, c2003

[15] Shuping Ran,*A model for web services discovery with QoS*, Source ACM SIGecom Exchanges archive,2003

[16] *Universal Description, Discovery and Integration*, Organization for Advancement of Structured Information System,2002, www.uddi.org/specification.html

[17] Y. Liu, AHH. Ngu, LZ. Zeng, *QoS Computation and Policing in Dynamic Web Service Selection*, WWW2004, May 17-20, New York, USA.