

2006

Using constraint hierarchies to support QoS-guided service composition

Y. Guan

University of Wollongong, yguan@uow.edu.au

Aditya K. Ghose

University of Wollongong, aditya@uow.edu.au

Z. Lu

University of Wollongong, lu@uow.edu.au

Publication Details

This paper was originally published as: Guan, Y, Ghose, A & Lu, Z, Using constraint hierarchies to support QoS-guided service composition, International Conference on Web Services (ICWS '06), Chicago, USA, September 2006, 743-752. Copyright IEEE 2006.

Using constraint hierarchies to support QoS-guided service composition

Abstract

A key impediment to the widespread adoption of web services is the relatively limited set of tools available to deal with Quality-of-Service (QoS) factors [12]. QoS factors pose several difficult challenges in how they may be articulated. While the functional requirements of a service can be represented as predicates to be satisfied by the target system, QoS factors are effectively statements of objectives to be maximized or minimized. QoS requirements occur naturally as local specifications of preference. Dealing with QoS factors is therefore a multi-objective optimization problem. In effect, these objectives are never fully satisfied, but satisfied to varying degrees. In evaluating alternative design decisions, we need to trade-off varying degrees of satisfaction of potentially mutually contradictory non-functional requirements. One key contribution of this work is the use of the constraint hierarchies framework from hierarchical constraint logic programming framework in dealing with Quality of Service(QoS) factors . We show how QoS factors can be formulated as soft constraints and how the machinery associated with constraint hierarchies can be used to evaluate the alternative trade-offs involved in seeking to satisfy a set of QoS factors that might pull in different directions. We apply also this approach to the problem of reasoning about web service selection and composition, and establish that significant value can be derived from such an exercise.

Disciplines

Physical Sciences and Mathematics

Publication Details

This paper was originally published as: Guan, Y, Ghose, A & Lu, Z, Using constraint hierarchies to support QoS-guided service composition, International Conference on Web Services (ICWS '06), Chicago, USA, September 2006, 743-752. Copyright IEEE 2006.

Using constraint hierarchies to support QoS-guided service composition

Ying Guan, Aditya K. Ghose, Zheng Lu
Decision Systems Laboratory
School of IT and Computer Science
University of Wollongong, Australia
{yg32, aditya, zl07}@uow.edu.au

Abstract

A key impediment to the widespread adoption of web services is the relatively limited set of tools available to deal with Quality-of-Service (QoS) factors [12]. QoS factors pose several difficult challenges in how they may be articulated. While the functional requirements of a service can be represented as predicates to be satisfied by the target system, QoS factors are effectively statements of objectives to be maximized or minimized. QoS requirements occur naturally as local specifications of preference. Dealing with QoS factors is therefore a multi-objective optimization problem. In effect, these objectives are never fully satisfied, but satisfied to varying degrees. In evaluating alternative design decisions, we need to trade-off varying degrees of satisfaction of potentially mutually contradictory non-functional requirements.

One key contribution of this work is the use of the constraint hierarchies framework from hierarchical constraint logic programming framework in dealing with Quality of Service(QoS) factors . We show how QoS factors can be formulated as soft constraints and how the machinery associated with constraint hierarchies can be used to evaluate the alternative trade-offs involved in seeking to satisfy a set of QoS factors that might pull in different directions. We apply also this approach to the problem of reasoning about web service selection and composition, and establish that significant value can be derived from such an exercise.

1 Introduction

A key impediment to the widespread adoption of web services is the relatively limited set of tools available to deal with Quality-of-Service (QoS) factors [12]. For instance, UDDI based look ups for web services are entirely based on the functional aspects of the desired services with quality factors playing no role. QoS factors encompass a wide range of non-functional attributes of a service such as

capability, performance, reliability, integrity, security etc. [9]. Although much progress has been made over the past several years, it is widely acknowledged that dealing with QoS factors for services remains an important open question.

QoS factors pose several difficult challenges in how they may be articulated. While the functional requirements of a service can be represented as predicates to be satisfied by the target system, QoS factors are effectively statements of objectives to be maximized or minimized and must be represented as such. Yet it is difficult and impractical to insist that QoS requirements be articulated by users as objective functions in the tradition of operations research techniques. QoS requirements occur naturally as local specifications of preference, and any robust approach to dealing with them must support such specifications. In evaluating alternative design decisions, we need to trade-off varying degrees of satisfaction of potentially mutually contradictory QoS factors. Dealing with QoS factors is therefore a multi-objective optimization problem.

QoS factors play a role in both service selection and composition. In both cases, there is usually no guarantee that a service can be found or composed that would exactly satisfy the given QoS requirements. In practical settings, we must deal with alternative that *deviate* from the given requirements. Our interest is in identifying those alternatives where the deviation is *minimal*. Ideally, we would like to have available a measure of *distance* between a potential service (or partially composed service) and a given set of QoS requirements. We would also like to be able to use this measure of distance to rule out less viable compositions early in the process. The framework we present in this paper addresses all of these requirements.

Our premise is that QoS factors can be modeled as soft constraints, and that the machinery of *constraint hierarchies* used in hierarchical constraint logic programming [15] can be brought to bear on the service composition problem (and also on the problem of service selection). We require that non-functional requirements (NFRS) and QoS factors ar-

ticated as inequalities relating key system parameters to thresholds on their values. A key challenge in dealing with quality factors is articulating them in terms of metrics, on which one could then apply thresholds or which one could seek to maximize or minimize. While some QoS factors lend themselves easily to such formulation, for others this is not entirely obvious. In [5], we list possible measures for some NFRs and QoS factors which do not have an obvious formulation as soft constraints, along the lines of the proposal in [2]. Those possible metrics would permit us to formulate constraint-style representations of quality factors. We acknowledge that such measures might not individually or jointly capture the complete real-life semantics of NFRs in question, and would thus constitute partial description of these NFRs. We also acknowledge that for some NFRs, such as maintainability and portability, well-understood measures probably do not exist. However, we believe that quantitative metrics for these can also be developed in the future, adding strength to our proposal.

In the QoSCH framework presented in this paper, QoS requirements for a service are represented as constraint hierarchies, which permit local specifications of optimization objectives as well as local specifications of preferences amongst objectives. The constraint hierarchies approach permits us to use a well-founded notion of distance, both for service composition and selection. We use this notion of distance to define a branch-and-bound procedure for service composition. We illustrate these with a detailed example.

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

2 Constraint Hierarchy

Constraint hierarchies (CHs) belong to traditional frameworks for the handling of over-constrained systems of constraints by specifying constraints with hierarchical preferences or strength. It allows one to specify not only hard constraints (the constraints that are required to hold), but also several preference levels of soft constraints (which violations are minimized level by level subsequently) at an arbitrary (finite) number of strengths [13].

To introduce the constraint hierarchies, we will use the definition of constraint hierarchies in [15]. A *constraint hierarchy* is a finite set of labeled constraints. A *labeled constraint* is a constraint labeled with a strength, written l_c 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 ($0 < k \leq l$, l is a constraint positive integer) as strength of a constraint instead of us-

ing symbols such as required, strong, medium and weak as the strength as in [15]. 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 θ . 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 constraints 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: $e("x = y", \theta) \equiv |\theta(x) - \theta(y)|$.

Constraint hierarchies 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 $1 < i < k$, $g(E(H_i\theta)) <_g g(E(H_i\sigma))$, and at level k , $g(E(H_k\theta)) <_g g(E(H_k\sigma))$. Currently, there are three groups of comparators: *global*, *local* and *regional comparators*. For a *local comparator*, each constraint is considered individually, for a *global comparator*, 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 $<>_g$ means equal). The *global comparator* includes *weighted-sum-better* (WSB), *worse-case-better* (WCB) and *least-squares-better* (LSB). In our model, we use the global metric comparator, which aggregate errors of violated constraints at each level.

3 Dealing with Web Service QoS factors using Constraint Hierarchies: The QoSCH framework

In this section, we will lay the groundwork for the application of constraint hierarchies in reasoning about web service QoS factors. As discussed earlier, we shall use such reasoning to support service selection and service composition.

Here we propose a general framework, the QoS via constraint Hierarchies (or QoSCH) framework, which is applicable for web service selection and web service composition. This framework is in a high-level abstraction without considering a particular language, algorithm, platform and other factors in the process of service selection and service composition. The architecture of QoSCH frame-

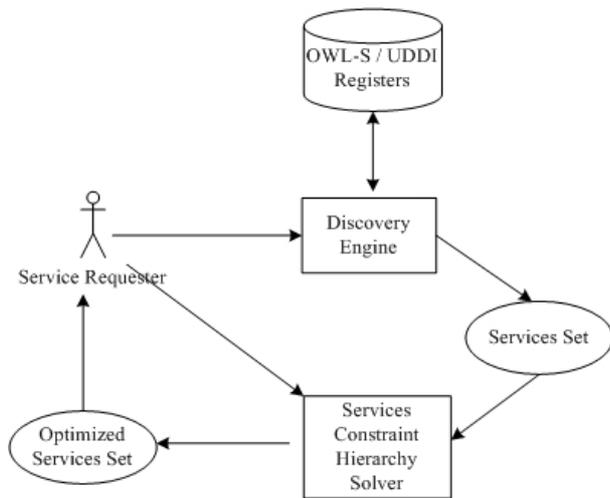


Figure 1. Architecture of QoSCH framework

work is illustrated in Figure 1. Different from standard web service architecture, we add a “Services Constraint hierarchy Solver” in the architecture of QoSCH framework. This component accepts constraint hierarchy from service provider and compute the performance distance from found services set to anticipated service. The output of this component is an optimized services set.

QoSCH framework that we presented in this section defines the following:

1. How service requirements (both functional and QoS) are defined.
2. The assumptions about services descriptions that must be satisfied for this framework to be applicable.
3. A measure of “distance” of a specific service from the “as-described”, i.e., the QoS requirements.
4. A \oplus operator which is associative to enable us to discuss service composition in abstract terms.
5. A \otimes operator to enable us to discuss the composition/aggregation of the QoS factors of individual service to obtain QoS descriptions of composite services, in abstract terms.

The QoSCH framework requires the following elements to be specified:

1. The functional service requirements. These can be represented in constraint-based form (which would be treated as hard constraints) or as assertions in some other formal or even semi-formal language.

2. QoS requirements. These will be represented as a constraint hierarchy, with each constraint relating a system parameter to a value, typically through an inequality for a parameter whose value one seeks to maximize, a constraint might constructed requiring that the parameter in question be assigned the highest possible value, viewed as a soft constraint, this would oblige the system to assign to this parameter as high value as possible, even if the highest value cannot be assigned. Similarly, the minimization objective for a parameter could be represented by a soft constraint that seeks to assign to this parameter the lowest possible value.
3. An instance of the \oplus operator referred to above. The most common instance of this operator is sequential composition, but parallel composition and other control structure may also be of interest.
4. An instance of the \otimes operator referred to above. In general this is a commutative, associative operator that seeks to combine QoS descriptions of individual services to a QoS descriptions the composition service. For example, if processing speed is the the only QoS factor of interest in a setting involving sequential composition of services, then arithmetic sum would be the appropriate instance of \otimes . More generally, the \otimes operator may be viewed as a vector of concrete operators, one for each QoS factor. If two QoS factors, processing speed and a reliability measure, were of interest in a sequential composition setting, then \otimes would be the vector [sum, min, max] where sum would be used to aggregate processing speed (for obvious reasons), min would be used to aggregate reliability (a sequence of services is as reliable as the least reliable service in the sequence) and max would be to aggregate response time (when compose two parallel services).
5. A machinery for measuring the distance of a given service from a service requirements specification. We shall discuss this machinery in the rest of this section.

The calculation of distance applies the variation of the constraint hierarchies in approach to obtain distance as defined below.

Let $\theta_c(x)$ represent the threshold obtained from an inequality constraint $c(x)$ on a variable x . thus if $c(x)$ is $x \leq 2$, then $\theta_c(x)$ is the value 2. We assume that σ is a value assignment to the parameters of interest. Thus $\sigma(x_i)$ represents the value assigned to x_i by this assignment function. An error function that determines how far a value assignment σ from a single variable x deviates from a unary constraint $c(x)$ is given by:

$$e(\theta_c(x), \sigma(x)) = |\theta_c(x) - \sigma(x)| / \theta_c(x) \times \text{notsatisfies}(c, \sigma)$$

where *notsatisfies* is a function defined as follows:

$$\text{notsatisfies}(c, \sigma) = \begin{cases} 0 & \text{if for all variable } x \text{ in the} \\ & \text{signature of } c, \sigma(x) \text{ satisfies } c \\ 1 & \text{otherwise} \end{cases}$$

Given a constraint hierarchy CH composed of n unary constraint $c_1(x_1), \dots, c_n(x_n)$ of varying weights the cumulative distance of a value assignment σ from CH is given by

$$d(CH, \sigma) = \sum_1^n e(\theta_{c_i}(x_i), \sigma(x_i)) \times w_i$$

where w_i denotes the weight of the constraint $c_i(x_i)$ in CH.

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

$$d = \sum_1^n |\theta(x_i) - \sigma(x_i)| / \theta(x_i) \times w_i = 1 \times 0 + 0.8 \times |0.02 - 0.025| / 0.02 + 0.5 \times |20 - 18| / 20 = 0.25$$

Note that if a valuation violate a required constraint, then it is immediately removed from consideration. If a service description does not refer to a system parameter that is used in a required constraint, then it is removed from consideration (i.e., distance set to ∞). If such a parameter does not appear in the required constraints, but elsewhere in the hierarchy, the service is not removed from consideration, but the error (in relation to each of the constraints that refer to this missing parameter) is set to the highest value (i.e., 1). If a service description refers to variables not referred to in the constraint hierarchy, then these can be used to resolve ties (in the case of service selection) between services that are the same distance from the constraint hierarchy.

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 might be in terms of QoS factors. Using the framework we have proposed, we can select a relatively optimal service from the set of available services. The selection steps are:

Step 1. Before selecting, we need to construct a QoSCH model to specify the service requirements. The \oplus and \otimes operator in the QoSCH model are irrelevant for service selection and can be ignored.

Step 2. Select those services that meet all the functional requirements.

Step 3. Calculate the distances from the constraint hierarchy of each web services selected in step 2.

Step 4. select the web service with minimum distance ($\min(d_i)$).

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 [11]. A composite web service is an aggregation of web services which interact with each other based on a process model [8]. Web service standards, such as UDDI, WSDL, SOAP, do not deal with the composition of existing services. The industry solution for web service composition is using WSDL and BPEL4WS (a business protocol specification language proposed by IBM and Microsoft). Many researchers have worked on this problem ([14, 8, 10]).

In this section, we propose a Branch and Bound Web Services Composition (BBWSC) technique that builds on the QoSCH model.

The branch and bound composition process consists of two steps:

1. Find the first composite service that meets all functional requirements and hard constraints. Let the distance from constraint hierarchy be d_i .
2. Try to construct another composite service for the same requirements. At each step in the composition process, compare the distance d of the partially composed service with d_i . If at any point $d > d_i$, then prune this branch.

Let S be a partially composed service, i.e., not all variables relating to QoS factors have been assigned values. Let $\text{Var}(S)$ denote the set of variable which have been assigned values in S . Let $\text{Project}(H, V)$ be the projection of constraint hierarchy H on the set of variables V . This is essentially the set of unary constraints in H that involve variables in V . We will refer to a function $\text{compositeQoS}(S)$ which takes a (possibly partial) composite service S and return a QoS description of S by aggregating the QoS description of component services of S , using the \otimes operator. Thus, if $S = s_1 \oplus s_2$ where s_1 and s_2 are atomic services and $\text{QoSDesc}(s_1)$ and $\text{QoSDesc}(s_2)$ provide QoS descriptions of s_1 and s_2 respectively, then $\text{compositeQoS}(S) = \text{QoSDesc}(s_1) \otimes \text{QoSDesc}(s_2)$.

BBWSC Algorithm:

```
S := null service;
d := ∞;
while alternative service composition exist do
  S := ∅;
  while ¬ complete(S) do
    S := S ⊕ s;
```


PS ₁	PS ₂	PS ₃	PS ₄
probability of detecting attack = 99.90%	probability of detecting attack = 99.92%	probability of detecting attack = 99.95%	probability of detecting attack = 99.00%
error rate < 0.0009%	error rate < 0.0011%	error rate < 0.0009%	error rate < 0.0010%
response time = 200 ms	response time = 250 ms	response time = 210 ms	response time = 220 ms
concurrent transactions = 1000	concurrent transactions = 800	concurrent transactions = 900	-
execution time = 500 ms	execution time = 500 ms	execution time = 550 ms	execution time = 400 ms
transaction cost = \$ 0.35/transaction	transaction cost = \$ 0.25/transaction	transaction cost = \$ 0.35/transaction	transaction cost = \$ 0.30/transaction
service cost = \$100/month	service cost = \$120/month	service cost = \$140/month	service cost = \$110/month

Figure 6. Valuations for constraints variables of Policyholder System

BPS ₁	BPS ₂
probability of detecting attack = 99.90%	probability of detecting attack = 99.92%
error rate < 0.0009%	error rate < 0.0010%
response time = 100 ms	response time = 100 ms
concurrent transactions = 1000	concurrent transactions = 1000
execution time = 200 ms	execution time = 200 ms
transaction cost = \$ 0.35/transaction	transaction cost = \$ 0.25/transaction
service cost = \$150/month	service cost = \$120/month

Figure 7. Valuations for constraints variables of Benefits plans systems

pose the constraint hierarchies for each service (Figure 8) and then calculate the distance from constraint hierarchy.

$$d_1 = 0.85, d_2 = 0.72. d_2 < d_1, d = 0.72.$$

The third composite service is $\{CS_1 \oplus PS_3 \oplus SPS_1\}$. When compose CS_1 and PS_3 , the distance is $0.845(> d)$. Therefore, this branch is pruned. For the same reason, branches $\{CS_1 \oplus PS_3 \oplus SPS_2\}$, $\{CS_2 \oplus PS_3 \oplus SPS_1\}$ and $\{CS_2 \oplus PS_3 \oplus SPS_2\}$ are also pruned. There are two composite service that meet the functional requirements, $\{CS_2 \oplus PS_1 \oplus SPS_1\}$ and $\{CS_2 \oplus PS_1 \oplus SPS_2\}$. The distances for these two services are 0.93 and 0.78 respectively. Finally, we get the best composition solution $\{CS_1 \oplus PS_1 \oplus SPS_2\}$ which has the smallest distance.

CS ₁ ⊕ PS ₁ ⊕ BPS ₁	CS ₁ ⊕ PS ₁ ⊕ BPS ₂
probability of detecting attack = 99.90%	probability of detecting attack = 99.90%
error rate < 0.0009%	error rate < 0.0009%
response time = 200 ms	response time = 200 ms
concurrent transactions = 1000	concurrent transactions = 1000
execution time = 1150 ms	execution time = 1150 ms
transaction cost = \$ 0.95/transaction	transaction cost = \$ 0.85/transaction
service cost = \$350/month	service cost = \$320/month

Figure 8. Valuations for constraints variables of Claim systems

Functional Requirements	Re-	Constraint Hierarchy of Quality Factors
Register		Constraints
Login/Logout		Strength
		probability of detecting attack ≥ 99.9%
Provide online claim form		error rate ≤ 0.001%
Process online claim request		response time ≤ 150 ms
		concurrent transactions ≥ 1000
		band-width ≤ 1024K
		service cost ≤ \$ 100/month

Figure 9. QoSCH model for Client system - Web porta

Functional Requirements	Re-	Constraint Hierarchy of Quality Factors
Verify request		Constraints
Queue request		Strength
		probability of detecting attack ≥ 99.9%
Invoke transaction		error rate ≤ 0.001%
Send result		response time ≤ 150 ms
		concurrent transactions ≥ 1000
		queue size ≥ 1000
		waiting time ≤ 12 h
		transaction cost ≤ \$ 0.75/transaction
		service cost ≤ \$ 300/month

Figure 10. QoSCH model for Enterprise service manager

Now, let us look at the composition of the whole system *Guardian Life Insurance System*. This system consists of four separated components, *Client system - Web portal*, *Enterprise service manager*, *Claim system* and *Data warehouse*. All these system are critical in the composed system. The QOSCH model for each system are listed in Figures 9, 10, 4 and 11. The QOSCH model for *Guardian Life Insurance System* is given in Figure 12.

There are two *Web portals*, two *Enterprise service managers* and three *Data warehouses* which meets requirements of that kind of system. For *Claim system*, we have got the best choose, the composition of *Claimsystem₁*, *Policyholdersystems₁* and *Benefitsplanssystems₂*.

Functional Requirements	Re-	Constraint Hierarchy of Quality Factors
Add record		Constraints
Delete record		Strength
		probability of detecting attack ≥ 99.9%
Query record		error rate ≤ 0.001%
Modify record		response time ≤ 100 ms
Backup database		concurrent transactions ≥ 10000
Build new database		execution time ≤ 800 ms
Generate report		service cost ≤ \$ 200/month

Figure 11. QoSCH model for Data warehouse

Functional Requirements	Re-	Constraint Hierarchy of Quality Factors	
Client system		Constraints	Strength
Enterprise service manager		probability of detecting attack \geq 99.9%	1
Claim system		error rate \leq 0.001%	1
Data warehouse		response time \leq 150 ms	0.9
		concurrent transactions \geq 1000	0.8
		queue size \geq 1000	0.8
		waiting time \leq 12 h	0.7
		execution time \leq 1500 ms	0.7
		band-with \leq 1024K	0.7
		transaction cost \leq \$ 1.50/transaction	0.6
		service cost \leq \$ 900/month	0.5

Figure 12. QoSCH model for Guardian Life Insurance System

Functional Requirements	Re-	Constraint Hierarchy of Quality Factors	
Client system		Constraints	Strength
Enterprise service manager		probability of detecting attack \geq 99.9%	1
Claim system		error rate \leq 0.001%	1
Data warehouse		response time \leq 150 ms	0.9
		concurrent transactions \geq 1000	0.8
		queue size \geq 1000	0.8
		waiting time \leq 12 h	0.7
		execution time \leq 1500 ms	0.7
		band-with \leq 1024K	0.7
		transaction cost \leq \$ 1.50/transaction	0.6
		service cost \leq \$ 900/month	0.5

Figure 13. Valuations for constraints variables of Web portal

ESM ₁	ESM ₂
probability of detecting attack = 99.90%	probability of detecting attack = 99.90%
error rate < 0.0009%	error rate < 0.0009%
response time = 200 ms	response time = 210 ms
concurrent transactions = 950	concurrent transactions = 800
queue size = 900	queue size = 800
waiting time = 14 h	waiting time = 16 h
transaction cost = \$ 0.95/transaction	transaction cost = \$ 1.00/transaction
service cost = \$320/month	service cost = \$350/month

Figure 14. Valuations for constraints variables of Enterprise service manager

DW ₁	DW ₂	DW ₃
probability of detecting attack = 99.90%	probability of detecting attack = 99.90%	probability of detecting attack = 99.00%
error rate < 0.0009%	error rate < 0.0010%	error rate < 0.0009%
response time = 200 ms	response time = 200 ms	response time = 200 ms
concurrent transactions = 9500	concurrent transactions = 9000	concurrent transactions = 10000
execution time = 1000 ms	execution time = 900 ms	execution time = 800 ms
service cost = \$250/month	service cost = \$250/month	service cost = \$220/month

Figure 15. Valuations for constraints variables of Data warehouse

DW ₁	DW ₂	DW ₃
probability of detecting attack = 99.90%	probability of detecting attack = 99.90%	probability of detecting attack = 99.00%
error rate < 0.0009%	error rate < 0.0010%	error rate < 0.0009%
response time = 200 ms	response time = 200 ms	response time = 200 ms
concurrent transactions = 9500	concurrent transactions = 9000	concurrent transactions = 10000
execution time = 1000 ms	execution time = 900 ms	execution time = 800 ms
service cost = \$250/month	service cost = \$250/month	service cost = \$220/month

Figure 16. Valuations for constraints variables of Guardian Life Insurance systems

The constrain hierarchy for those available systems are listed in Figure 13, 14, 8 and 15. Because DW_3 does not satisfy constraints in *require* level, then those branches that contain DW_3 are pruned. According to BBWSC, we found the first composite service $\{WP_1 \oplus ESM_1 \oplus CS, DW_1\}$ that meets all the functional requirements and hard constraints. Compose the constraint hierarchies for each service (Figure 16) and then calculate the distance from constraint hierarchy.

$$d = d_1 = \sum_1^n |\theta(x_i) - \sigma(x_i)| / \theta(x_i) \times w_i = 1.13.$$

The second composite service for the same requirements is $\{WP_1 \oplus ESM_1 \oplus CS, DW_2\}$. The composite constrain hierarchy is in Figure 16. the distance from constraint hierarchy is $d_2 = 1.02$. Because $d_2 < d_1$, $d = 1.02$.

The third composite service is $\{WP_1 \oplus ESM_2 \oplus CS \oplus DW_1\}$. When compose WP_1 , ESM_2 and CS , the distance is $1.07 (> d)$. Therefore, this branch is pruned. For the same reason, branches $\{WP_1 \oplus ESM_2 \oplus CS \oplus DW_2\}$, $\{WP_2 \oplus ESM_1 \oplus CS \oplus DW_1\}$, $\{WP_2 \oplus ESM_1 \oplus CS \oplus DW_2\}$, $\{WP_2 \oplus ESM_2 \oplus CS \oplus DW_1\}$ and $\{WP_2 \oplus ESM_2 \oplus CS \oplus DW_2\}$ are also pruned. Finally, we get the best composition solution $\{WP_1 \oplus ESM_1 \oplus CS \oplus DW_2\}$ ($\{WP_1 \oplus ESM_1 \oplus CS_1 \oplus PS_1 \oplus SPS_2 \oplus DW_2\}$) which has the smallest distance.

4 Related Work

Functionality and non-functional properties are two essential factors to each web service. Functionality is used to measure whether this web service meets all the functional requirements of an anticipated web service, while non-functional properties are qualified to evaluate the performance of the web service. This has been viewed as a sufficient 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 as well as selection and composition of web service. In [12], the author proposed a QoS model which offers a QoS certified 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 [16]. In [8], authors proposed a global planning approach to optimally select component services during the execution of a composite service. This proposed approach is quality-driven and by using Multiple Attribute Decision Making (MADM) [7] approach select optimal execution plan. This approach 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 based web services composition, improve the composition efficiency in great extent.

There are many work have been done to develop language for specifying the QoS factors of web services. OWL-S ontology [1] is the only popular web service composition approach that support the description of non-functional requirements parameters. In [3], an ontology QoSOnt was proposed as an extension to OWL-S and works in symbiosis with OWL-S. It is designed to provide a common QoS conceptualisation for services provider, services requesters or a third party intermediary. We could use either of these two ontology for the specifications of our QoSCH framework.

5 Conclusion and Future Work

In this paper we proposed to use the hierarchical constraint logic programming framework in dealing with QoS factors. 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. Our focus is on showing how QoS factors can be formulated as soft constraints and how the machinery associated with constraint hierarchies can be used to evaluate the alternative trade-offs involved in seeking to satisfy a set of QoS factors that might pull in different directions. Moreover, we apply this approach to web service selection and

web service composition. Our larger project seeks to deploy the full capability of the HCLP framework in dealing with non-functional requirements. In the current work, we only focus on the constraint hierarchy component of framework. There is one limitation of our currently work, that is, QoSCH model is based on a simplified version of constraint hierarchy framework, not a traditional constraint hierarchy framework. We plan to extend QoSCH framework based on traditional traditional constraint hierarchy framework. In the future work, we also need to choose a suitable Web Ontology Language for specifications of constraint hierarchy of QoS factors.

References

- [1] Owl-s home page. <http://www.daml.org/services/owl-s/>. 2003.
- [2] P. C. Bass Len and R. Kazman. *Software architecture in practice*. Boston : Addison-Wesley, 2003.
- [3] R. L. Glen Dobson and b. . Ian Sommerville. Qosont: a qos ontology for service-centric systems.
- [4] G. Gruman. Soa ensures guardian gets it right. pages 44–45, May 2005.
- [5] Y. Guan and A. K. Ghose. Dealing with web service qos factors using constraint hierarchies. In *Proceedings of 17th International Conference on Software Engineering and Knowledge Engineering (SEKE-2005)*, 2005.
- [6] H. Hosobe. *A foundation of Solution Methods for Constraint Hierarchies*. Kluwer Academic Publishers, 2003.
- [7] M. Kksalan and S. Zions. *Multiple Criteria Decision Making in the New Millennium*.
- [8] M. D. J. K. Liangzhao Zeng, Boualem Benatallah and Q. Z. Sheng. Quality driven web services composition. In *Proceedings of the twelfth international conference on World Wide Web*, pages 411–421, 2003.
- [9] D. A. Menasce. Qos issues in web services. *IEEE Internet Computing*, 6:72–75, 2002.
- [10] G. R.-G. Michael C. Jaeger and G. Mhl. Qos aggregation for web service composition using workflow patterns. In *Proceedings of Eighth IEEE International Enterprise Distributed Object Computing Conference(EDOC'04)*, 2004.
- [11] M. R. B. Paulo F. Pires and M. Mattoso. Building reliable web services composition. In *Lecture Notes In Computer Science*, volume 2593, pages 59–72, 2002.
- [12] S. Ran. A model for web services discovery with qos. *ACM SIGecom Exchanges*, 4:1–10, 2003.
- [13] H. Rudov. *Constraint Satisfaction with Preferences*, PhD thesis. Faculty of Informatics, Masaryk University, 2001.
- [14] B. Srivastava and J. Koehler. Web service composition current solutions and open problems. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, 2003.
- [15] M. Wilson. *Hierarchical Constraint Logic Programming*, PhD Thesis. University of Washington, May 1993.
- [16] N. Y. Liu, AHH and L. Zeng. Qos computation and policing in dynamic web service selection. In *Proceeding of International World Wide Web Conference on Alternate track papers & posters*, pages 66–73, 2005.