Dealing with Quality Tradeoffs during Service Selection

Caroline Herssens
PReCISE Research Center
Louvain School of Management
Université catholique de Louvain
caroline.herssens@uclouvain.be

Ivan J. Jureta and Stéphane Faulkner
PReCISE Research Center
Louvain School of Management
University of Namur
{ivan.jureta, stephane.faulkner}@fundp.ac.be

Abstract

In a service-oriented system (SoS) service requests define tasks to execute and quality of service (QoS) criteria to optimize. A service request is submitted to an automated service selector in the SoS, which allocates tasks to those service that, together, can "best" satisfy the given QoS criteria. When the selector cannot optimize simultaneously the given QoS criteria, users need to specify priorities over the said criteria. Accounting for users' QoS priorities is therefore necessary during service selection. Once specified by the requester, quality properties will be used by the selector to lead autonomic optimization of the service selection process. We outline and test a selection approach that accommodates priorities and that is based on available Multi Criteria Decision Making techniques.

1. Introduction

Engineering and managing the operation of increasingly complex information systems is a key challenge in computing (e.g., [14]). It is now widely acknowledged that degrees of automation needed in response cannot be achieved without open, distributed, interoperable, and modular systems capable of dynamic adaptation to changing operating conditions. Among the various approaches to building such systems, service-orientation stands out in terms of its reliance on the World Wide Web infrastructure, availability of standards for describing and enabling interaction between services, attention to interoperability, and uptake in industry.

A *service* is a self-describing and self-contained modular application designed to execute a well-delimited task, and that can be described, published, located, and invoked over a network [19, 26]. A *web service* is a service made available on the World Wide Web. Services are offered by *service providers*, i.e., organizations that ensure service implementations, advertise service descriptions, and provide related technical and business support. A service-oriented system (SoS) incorporates *service selectors*. A *service selector* receives *service requests* from human users or other systems, then automatically discovers, selects, and coordinates the execution of services so as to fulfill given service requests (e.g., [11, 18, 22, 28, 27]). A service request usually describes *what* is to be done and *how well*; that is, the tasks that stakeholders expect the system to execute and/or achieve, and the constraints on Quality of Service measures to meet while performing the tasks.

Service orientation is intended to enable large scale systems. Many competing services are therefore available to perform the same tasks. In such a setting, the service selector aims to select the set of services that optimally satisfies the QoS considerations laid out in the request, and this relative to alternative sets of services that can perform the same tasks. *QoS considerations guide the selection process in presence of many competing services* [20, 24]. *Appropriate comparison of alternative services and subsequent selection requires expressive QoS models and mechanisms for managing conflicts that appear in stakeholders' service requests* (e.g., two QoS criteria are selected for optimization, although they are conflicting). Autonomic computing cannot be realized through service-orientation without such expressive QoS models and comparison mechanisms.

The QoS model is used in an SoS to make explicit the various QoS dimensions and characteristics that can be used to specify QoS considerations in service requests and measure them at runtime on each service. Any such model is therefore used (i) by service requesters to specify the expected quality levels of service delivery; (ii) by service providers to advertise quality levels that their services achieve; and (iii) by service selectors when selecting among alternative services those that are to participate in a service composition. Given a QoS model, a service request may specify that several QoS dimensions be optimized. If the said dimensions involve tradeoffs, they cannot be optimized and additional information is needed in a service request. Namely, we expect the stakeholders to indicate the

978-0-7695-3175-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICAC.2008.8

priority over the said QoS dimensions so that an order of importance is established, and subsequently used to guide optimization. Though the need for priorities is evident, very limited work has been performed to deal with them during service selection.

We cannot reasonably expect the services to indefinetly maintain same QoS levels. Moreover, new services emerge and old ones may become unavailable. Selection therefore must account for the variations in QoS levels at runtime to enable self-managed processes. Different classes of approaches to service selection acknowledge this difficulty [9, 36]. Most service selection algorithms introduce quality dimensions either as objective function to be minimized (e.g., execution duration) or maximized (e.g., availability) or either as constraints to be satisfied [9, 24, 36]. They unfortunately do not allow for priorities among QoS dimensions. Multi-criteria decision making (MCDM) techniques allow priorities to be accounted for during decision making. Proposals that deploy MCDM for service selection are available [6, 30, 33, 36]. Their shared limitation is that they cannot accommodate variation of QoS levels at individual services observed at runtime (that is, they assume known and stable QoS levels).

Contributions. Instead of developing a particular service selection procedure which accommodates priorities between QoS considerations, we take a different approach: we provide an extension compatible with (i.e., that can be used with) available selection approaches. Our approach is based on the premise that service-oriented systems for autonomic computing operate in a setting in which QoS levels vary and are observed at runtime. The approach is based on a specific class of MCDM techniques: the outranking methods. The approach enables the definition of a *global priority constraint* to be used as an ordinary constraint in a service selection algorithm. We do not ask for a specific class of service selection algorithms; any algorithm which proceeds to select the optimal services and accounts for QoS considerations can be used in conjunction with the present proposal. This is for instance the case with the reinforcement learning algorithm we suggested elsewhere [11, 10]. The constraint is relevant because:

- it allows priorities to be accounted for during service selection in algorithms that originally cannot accommodate priorities;
- it can be integrated with various available service selection algorithms, and regardless of their specific optimization functions;
- it enables automatic optimization of user preferences by the service selector.

The paper makes an additional, conceptual contribution. Our approach requires richer QoS models than those available in the literature, so that we introduce an extension to the QoS metamodel of the Unified Modeling Language. The model will facilitate the representation of priorities.

Organization. We first present the QoS model we subsequently use in service selection (§2). We then show how to specify priorities over QoS considerations, integrate priorities with the QoS model, and explain the conversion of priorities to weights (§3). Next, we describe our global priority constraint and its definition through outranking methods (§4). We illustrate the use of the global priority constraint (§5). We close the paper by reviewing related efforts (§6) and summarizing our conclusions and directions for future work (§7).

2. QoS Modeling

As we noted above, choice among competing services (that is, services which can perform the same tasks) needs to be guided by the services' nonfunctional (i.e., quality) characteristics, expressed in the form of QoS criteria and constraints. We need a QoS model to show how priorities can be accounted for in service selection. Among the various QoS models (e.g., [3, 12, 13, 16, 35, 37]) we use herein the UML QoS Framework [23]. It provides a metamodel which is instantiated to obtain a QoS model. We do not go into the detail of this metamodel for the available literature already does this; instead, we only mention parts thereof relevant for the present discussion. The mentioned parts are relevant because the UML QoS Framework metamodel does not accommodate priorities. We thus extend the metamodel in a simple way later on in order to allow priorities. Two parts of the metamodel relevant in such a setting are the following:

- **QoS Characteristic** A QoS Characteristic is a description for some quality consideration, such as, e.g.: latency, availability, reliability, capability. A characteristic is quantified by means of specific parameters and methods. These concepts are provided by the metaclass *QoS Parameter*. Extensions and specializations of such elements are available with the sub-parent self-relation. A characteristic has the ability to be derived into various other characteristics as suggested by the templates-derivations self-relation.
- **QoS Dimension** A QoS Dimension specifies a measure that quantifies a QoS characteristic. The attribute *direction* defines the direction (increasing, decreasing) in which it is desired that the value of the QoS Dimension moves. *Unit* and *statistical qualifier* attributes

specify, respectively, the unit for the value dimension and the type of the statistical qualifier; e.g.: maximum value, minimum value, range, mean, frequency, distribution, etc.

These two parts of the UML QoS metamodel are shown in Figure 1, which also shows how the metamodel is extended to accommodate priorities. The attributes *type* and *parameters* have been added to describe within-criterion information as explained in Subsection 4.1. The extension is explained in the following section.

3. Priorities over QoS

3.1. Adding priorities to the UML QoS framework metamodel

Given that we use the UML QoS Framework metamodel, we can define priority orders over distinct QoS Characteristics and over distinct QoS Dimensions. There is no interest in defining mixed priority orders such as, e.g., priority between a QoS dimension and a QoS characteristic. For two distinct QoS Dimensions $\mathbf{d}_i$ and $\mathbf{d}_j$, we write $\mathbf{d}_i \succeq_d \mathbf{d}_j$ to express that improving the value of $\mathbf{d}_i$ is at least as important as improving the value of $\mathbf{d}_j$. For two distinct QoS Characteristics $\mathbf{c}_i$ and $\mathbf{c}_j$, we write $\mathbf{c}_i \succeq_c \mathbf{c}_j$ and interpret it as follows: improving any of the quality dimensions defining $\mathbf{c}_i$ is at least as important as improving any of the quality dimensions defining $\mathbf{c}_j$. Both $\succeq_d$ and $\succeq_c$ are transitive and strict priority is defined as usual (i.e., $x \succ y \equiv x \succeq y \wedge \neg(y \succeq x)$).

If, for example we need to express in a service request that it is strictly more important to optimize the security QoS Characteristic than the performance QoS Characteristic, we write the following: Security $\succ_c$ Performance.

To accommodate the priority orders between dimensions and between characteristics, we extend the UML QoS Framework metamodel with the following metaclasses (shown in Figure 1):

- **QoS Priority** This class is used to express *rules* that define priorities over characteristics or dimensions. These rules determine the order in which dimensions or characteristics are considered for improvement/optimization when tradeoffs arise. A rule expresses an order relation between elements. The *blank criteria* attribute is used to express the relative importance of the priority as presented in Subsection 3.2.

- **QoS DimPriority** and **QoS CharactPriority** These classes are specializations of QoS Priority for priorities over, respectively, characteristics and dimensions.

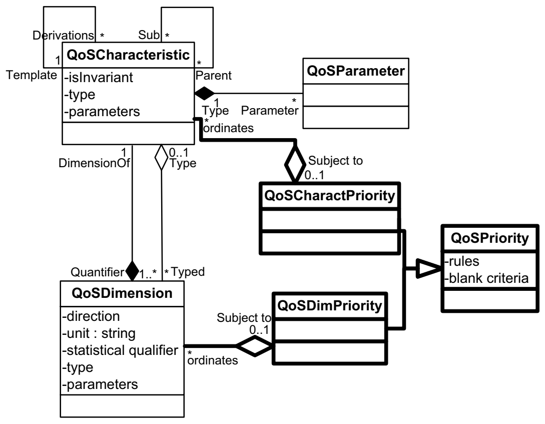


Figure 1. Part of the UML QoS Framework metamodel with priorities extensions in bold

3.2. Using weights to express priorities

Priorities as expressed above cannot be directly used in a service selection method. To enable an autonomic process given the priority orders, we need to associate weights with each QoS Dimension or QoS Characteristic. To see how weights can be associated to QoS Dimensions or Characteristics, we need to answer how the weights are aggregated in a service selection procedure. Two aggregation approaches are available: the compensatory and the non-compensatory.

Compensatory approaches such as, e.g., the Analytic Hierarchy Process (AHP) [29] used for e-business process composition [30], present some disadvantages. Weights which appear in compensatory approaches amount to being substitution rates, allowing differences in priorities as they relate to different criteria (in the present terminology, QoS Dimensions or Characteristics) to be expressed on the same scale. They therefore do not characterize the intrinsic relative importance of the attributes.

Non-compensatory approaches refer to weights simply as a measure of the relative importance of the criteria involved. Non-compensatory weighting can be compared to the number of votes given to a candidate in a voting procedure, with the final tally indicating the relative importance of each criterion 'candidate' [34]. Different methods of criterion weighting exist, among them, those proposed by Hokkanen and Salinen [8], Simos [31] or Mousseau [21]. In order to make a compromise between usability and expressiveness for specifying priorities over QoS Dimensions and Characteristics, we choose to use an extended version of the Simos weighting procedure described by Figueira and Roy in [4].

In the original Simos procedure, criteria are put in the order of importance that the decision-maker considers appropriate. The decision-maker can also add 'blank' criteria to reinforce rank differences. Criteria with the relative same importance can be put on the same rank. The lowest order rank is assigned to the number '1' and the decision-maker then proceeds upwards. The rankings are increasingly unequal as more blank criteria are used. The revised procedure introduces a new kind of information: the decision-maker is asked *how many times the last criterion is more important than the first in the ranking*. Moreover, drawbacks of the subsets of ex aequo of the original Simos procedure are eliminated and ameliorations processes the rounding off of the numerical values in an optimal way. Information concerning the number of blank criteria is specified in the QoS Priority metaclass introduced in Subsection 3.1.

4. Using Outranking Methods to Define the Global Priority Constraint

Instead of developing a particular service selection procedure which accommodates priorities between QoS Dimensions and Characteristics, we take a different approach: we provide an extension to available selection approaches. The extension enables the definition of a global priority constraint to be used as an ordinary constraint in the service selection problem. We do not ask for a specific class of service selection algorithms; any algorithm which proceeds to select the optimal services and accounts for QoS Dimensions and/or Characteristics can be used in conjunction with the present proposal. This is for instance the case with the reinforcement learning algorithm we suggested elsewhere [11].

The purpose of the global priority constraint is to define the relative priority of a set of QoS Dimensions (and thereby Characteristics). Weights are assumed specified through the improved Simos weighting procedure. Among the various multi-criteria decision making methods, we choose outranking methods to define the global priority constraint. Outranking methods start from a set of alternatives (alternative selected individual services) and a set of criteria (QoS Dimensions or QoS Characteristics), in order to evaluate alternatives. The methods vary in the amount and kind of additional information required; this additional information may be, e.g., the decision-maker's preferences (herein interpreted as priorities), domain- and problem-specific information, and so on.

An outranking relation is a binary relation *S* defined on the set of potential choices *A* such that $a_i S a_j$ if there is enough information (i.e., arguments) to decide that a_i is at least as good a choice as a_j, whereby there is no information (i.e., counterarguments) that refutes that conclusion. Common classes of outranking methods are ELECTRE and PROMETHEE methods. The PROMETHEE method has the advantage to consider explicitly information between criteria and information within each criterion. Inter-criteria information concerns mainly weights attributed to each criterion (i.e., priority assigned to each QoS attribute).

4.1. Promethee

The PROMETHEE [1] class of outranking methods performs pairwise comparisons of alternatives by considering the deviation between the evaluations of the alternatives. The more significant the deviation, the higher the preference. We interpret the higher preference as higher priority herein.

The result of the pairwise comparison for a criterion to maximize is given by:

$$P_j(a,b) = F_j[d_j(a,b)] \forall a,b \in A \tag{1}$$

where

$$d_j(a,b) = g_j(a) - g_j(b) \tag{2}$$

and for which

$$0 \leq P_j(a,b) \leq 1 \tag{3}$$

Where:

- $P_j(a,b)$ is the priority of the selection of some service (choice a) over the selection of another service (choice b) over the QoS Dimension j;
- $d_j(a,b)$ is the deviation between the choice a and choice b over the QoS Dimension j;
- $g_j(a)$ is the score of the service a over the QoS Dimension j;
- F_j is the function giving the within-criterion information associated to the QoS Dimension j.

The result of a pairwise comparison on a QoS Dimension to minimize is such that:

$$P_j(a,b) = F_j[-d_j(a,b)] \tag{4}$$

Depending on the inherent characteristics of a given QoS Dimension, the user of the approach can use one of six kinds of functions for within-criterion information. These are overviewed in details in [1, 34], each type necessitates some particular parameters: type 1 is referred to as immediate preference; type 2 introduces an indifference threshold; type 3 increases continuously until this indifference threshold; type 4 comprises an indifference and a preference thresholds; type 5 increases continuously between indifference and preference thresholds and; type 6 follows a Gaussian law with a fixed standard deviation. These types and their related parameters are specified with help of attributes *type* and *parameters* of the QoS Dimension and QoS Characteristic metaclasses introduced in Section 2.

In order to establish the ranking relation between alternatives, we first need to define aggregated preference indices and outranking flows.

- **Aggregated preference indices**
  The *aggregated preference indices* are used to express to what degree is the choice a preferred to the choice b over all considered QoS Dimensions ($\pi(a, b)$) and inversely, to what degree is the choice b preferred to the choice a over all considered QoS Dimensions ($\pi(b, a)$). Most of time, a will be of higher priority to b for some QoS Dimensions and b will be of higher priority to a for others. Thereof, $\pi(a, b)$ and $\pi(b, a)$ are usually positive.

  $\pi(a, b)$ and $\pi(b, a)$ are defined by:

$$\begin{cases} \pi(a,b) = \sum_{j=1}^{k} P_j(a,b) w_j \\ \pi(b,a) = \sum_{j=1}^{k} P_j(b,a) w_j \end{cases} \tag{5}$$

  where w_j is the weight associated to the QoS Dimension j and k is the number of distinct QoS Dimensions.

  The Promethee method can also be used with QoS Characteristics, depending on the information available to the user.

- **Outranking flows**
  The *outranking flows* determine how each choice a is facing the $n - 1$ other possible choices in A. The positive outranking flow ($\phi^+(a)$) expresses how an alternative a is outranking all the others, the higher its value, the better the alternative. The negative outranking ($\phi^-(a)$) expresses how an alternative a is outranked by $n - 1$ other alternatives. The lower its value is, the better is the alternative.

  $\phi^+(a)$ and $\phi^-(a)$ are defined by:

$$\begin{cases} \phi^+(a) = \frac{1}{n-1} \sum_{x \in A} \pi(a,x) \\ \phi^-(a) = \frac{1}{n-1} \sum_{x \in A} \pi(x,a) \end{cases} \tag{6}$$

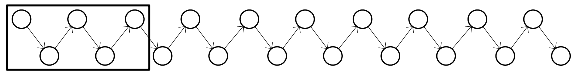
Once these outranking flows have been determined, several ways of ranking are available. PROMETHEE I proposes a partial ranking of alternatives authorizing equalities over alternatives while PROMETHEE II provides a complete ranking of alternatives. To define our global constraint on available services, complete ranking offers more information than partial one, so we choose to use PROMETHEE II. The complete ranking of PROMETHEE II is defined by:

$$\phi(a) = \phi^+(a) - \phi^-(a) \tag{7}$$

4.2. Using Promethee to define the global priority constraint

The PROMETHEE II method offers a complete ranking over alternatives while considering multiple criteria weighted according to their importance. Rather than selecting the better service ranked with the help of the outranking method, our aim is to fix a constraint applicable to multiple QoS selection and composition models. This way, different algorithms and optimization function may be used together with the defined constraints (as in, e.g., [11]). Classical approaches limit service selection by constraints defined a priori without considering actual QoS values. A such approach risks that constraints are satisfied or rejected by all candidate services. The utilization of PROMETHEE furnishes a ranking of services based on observed values of QoS Dimensions. The result of this ranking can be represented on an oriented graph. Once this ranking is processed, the constraint determine an acceptance level over this ranking. For example, only the first $x\%$ of services ranked by the promethee method may be selected as illustrated on oriented graph in Figure 2.

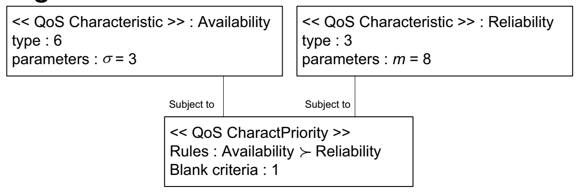
Figure 2. Selection given a ranking



5. Practical Example

The first step is to determine which criteria will account for the selection and represent them with the QoS metamodel. Among all possibilities, we select five QoS Characteristics that will induce our selection. These are: availability, latency, reliability, reputation and security. The next step is to define priorities with help of the QoS metamodel and to affect weights to define priorities. Small parts or our instance of the metamodel are illustrated in Figure 3. Third, we will define intra-criterion information and execute the algorithm to establish the ranking. Finally, we will construct the selection constraint on available alternative services.

Figure 3. Instantiation of the QoS metamodel



<< QoS Characteristic >> : Availability
type : 6
parameters : σ = 3
<< QoS Characteristic >> : Reliability
type : 3
parameters : m = 8
Subject to
Subject to
<< QoS CharactPriority >>
Rules : Availability ≻ Reliability
Blank criteria : 1

5.1. Determining Weights

To set weights, we need first to establish an order relation on criteria and, eventually, introduce *blank criteria* adding information about the relative importance between two successive criteria. With the revised Simos procedure, we also require to express z, the importance factor between the first and the last criterion.

We choose to establish the following priority ordering:

$$\begin{cases} reputation \succ_c security \\ security \succ_c latency \\ latency =_c availability \\ availability \succ_c reliability \end{cases} \tag{8}$$

Recall that we defined the priority relation as transitive earlier. Also, we shall for simplicity not go into the detail of individual QoS Characteristics (we do not make explicit how they are measured, that is, do not reveal the relevant QoS Dimensions).

We add a blank criterion between the rank formed by availability and latency and the rank of reputation to express a more important difference than between other criteria as illustrated in Figure 3. We give to z, the factor determining how many times the last criterion is more important than the first one in the ranking, the value 6. The Table 1 presents the non normalized weights obtained with the procedure described in [4].

Once we have obtained the non-normalized weights, we need now to normalize them. The revised procedure uses a new technique to normalize the weights while minimizing the rounding off efforts. Computation results of normalization are displayed in Table 2. Once weights are normalized, we need to determine which ones will be rounded upwards to have the sum of normalized weights equal to 100. To that end, dysfunctions of relative error of rounding up (d_i) and down ($\bar{d_i}$) are calculated.

We determine that criteria to be rounded up are reliability and security while other criteria (availability, latency, reputation) are rounded down. The last column of Table 2 give us final weights (k_i) representing priorities of quality attributes considered.

5.2. Constructing the Outranking Relation

To fix our outranking relation, we still need within-criteria information. We must determine the type of each criterion (here, QoS Characteristic) and their respective parameters. Types and parameters are specified with the QoS metamodel as illustrated in Figure 3 for availability and reliability characteristics. Our choices are presented in Table 3. Optional parameters are added when required; l is the preference threshold; m is the strict preference threshold, and σ determines the inflection point of the Gaussian criterion.

Table 4. Characteristics of available services

Provider	Availability	Latency	Reliability	Reputation	Security
1	78	410	83	92	8
2	71	380	78	91	7
3	87	455	74	95	6
4	57	240	86	76	9
5	82	380	67	86	7
6	92	520	87	90	6
7	74	450	85	91	7
8	86	400	76	92	6
9	76	380	82	89	9
10	78	390	87	90	8

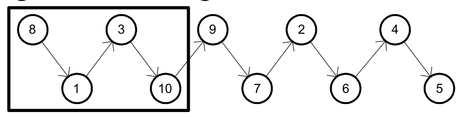
We can now execute the outranking method on services supplied by 10 different providers. Their QoS are mentioned in Table 4.

With the performance of providers in Table 4, the intra-criterion information resumed in Table 3 and the weights available in Table 2, we can calculate positive and negative outranking flows. Next, we can compute the complete ranking; these are given in Table 5.

Table 5. Outranking flows

Provider	ϕ^+	ϕ^-	ϕ
1	53,09	32,46	20,63
2	38,12	47,92	-9,80
3	49,90	37,61	12,29
4	37,30	58,47	-21,17
5	35,40	65,05	-29,65
6	33,08	44,11	-11,03
7	28,95	37,10	-8,15
8	60,61	26,71	33,90
9	45,48	42,63	2,85
10	46,17	36,03	10,14

Figure 4. Ranking of available services



The flow result gives us the ranking of available services. This ranking is illustrated on Figure 4.

5.3. Defining the Constraint

The aim of our model is to automatically provide a global constraint usable with other algorithms or in composition problems. This constraint is built from the ranking and imposes the selection of a service from the x first percent of considered services. This way, it restricts the set of possible alternatives by rejecting the least competitive services. In

Table 1. Non-normalized weights for $z = 6$

Rank r	Criteria in the rank r	Number of white cards according to rank r, e'_r	e_r	Non-normalized weights $k(r)$	Total
1	reliability	0	1	1.00	1.00 * 1 = 1.00
2	security	0	1	2.25	2.25 * 1 = 2.25
3	availability, latency	1	2	3.50	3.50 * 2 = 7.00
4	reputation	...	...	6.00	6.00 * 1 = 6.00
sum	5	1	4	...	16.25

Table 2. Determining the normalized weights of each criterion for $w = 1$ and $z = 6$

Rank	Criteria	Normalized weights k_i^*	Normalized weights k_i''	Ratio d_i	Ratio $\bar{d}_i$	Normalized weights k_i
1	reliability	6.1538	6.1	0.00750755	0.00874256	6.2
2	security	13.8461	13.8	0.00389279	0.00332945	13.9
3	availability	21.5384	21.5	0.00241205	0.00150362	21.5
3	latency	21.5384	21.5	0.00241205	0.00150362	21.5
4	reputation	36.9230	36.9	0.00208542	0.00062291	36.9
sum	5	100	99.8	...	...	100

Table 3. Within criteria information

Criteria	Unit	Type	Direction	Parameter	Parameter's value
Availability	%	Criterion with Linear preference	increasing	m	8
Latency	ms	Quasi-Criterion	decreasing	l	45
Reliability	%	Gaussian Criterion	increasing	σ	3
Reputation	%	Criterion with Linear Preference	increasing	m	4
Security	Level (1 to 10)	Usual Criterion	increasing	-	-

this example, we choose to accept only the first 40% of services outranked as illustrated in the box on Figure 4. This global priority constraint admits only the best services in accordance with our parameter choices.

6. Related Work

In this section, we review previous efforts related to the service selection problem. We focus our selection approach on quality properties of web services that have been widely introduced in the context of service-oriented computing. O'Sullivan and colleagues [24] suggest a complete description of nonfunctional properties of services and review their use in discovery, substitution, composition and management. They consider QoS to be constraints over the functionality of a service. Menascé [20] considers quality as a combination of several qualities and properties of a service and discuss about evaluation of QoS measures from the user's and the provider's perspectives.

Services selection is an emerging issue giving rise to various issues, from algorithms of selection to architectures supporting QoS models allowing comparisons over alternatives [25]. Tian and colleagues [32] outline an approach that enables the selection of appropriate services based on QoS requirements. To that purpose, they propose an architecture in which services providers offer different classes of the same service with different levels of QoS and price.Services selection problem has been introduced in two areas of research: the service selection issue and the selection of services integrated in a services composition.

6.1. Service selection

We review here some of the most closely related efforts. In contrast to our proposal, the approaches highlighted below do not provide an explicit approach to accommodate and compute priorities over QoS. Moreover, the aim of these efforts is to select the best alternative from available alternatives by fixing a priori constraints on QoS; they cannot be combined with other available algorithms for service selection. Liu, Ngu and Zeng [15] outline an approach that establishes a QoS model to define non-functional properties. The model covers some generic QoS characteristics and allows extension to other domain-specific quality criteria. Once domain-specific QoS are fixed, a method is applied to establish a ranking of service alternatives. The web service selection algorithm determines which service is selected based on end user's constraints. Available services and their respective values on criteria are inserted in a matrix. The matrix is then normalized in order to allow for a uniform measurement of service qualities independently of unit specifics. As different users have different QoS expectations, the model then weighs QoS characteristics to establish the ranking. The normalization and the weighting of their proposal lead to a compensatory approach that does not characterize the intrinsic relative importance of criteria involved. Maximilien and Singh [17] propose a multi-agent system providing a dynamic and self-adjusting selection of services based on trust. This framework uses a selection algorithm based on trust which does not authorize to link weights and consequently, priorities. The trust of alternatives is based on provider advertisement of particular qualities and the consumer preferences for those qualities. These qualities are normalized to enable their comparison and aggregation into a single value. Tong and Zhang [33] suggest a fuzzy multi attribute decision making algorithm to solve the service selection problem. They account for five weighted quality criteria for simple services. Quality vectors are first normalized and best and worst services are subsequently identified. Services closest to the best and farthest from the worst solution are selected. Contrary of our approach, thei proposal is based on the best and worst solution possible and is not adjusted to performance of existing alternatives. Casati and colleagues [2] outline a data-mining approach to service selection. They analyze past executions of services and build a set of context-sensitive service selection models to be applied at each stage of service execution. They account for use requirements in their approach but they do not allow to link priorities over considered quality properties. Ghosh and colleagues [5] provide an approach to QoS optimization that may be applicable to service selection. QoS optimization is performed dynamically and in real-time, whereby the problem is that of allocating resources to specific tasks, and therefore similar to the service selection problem. The user is asked to provide, for each task, a utility function that maps each point in the quality space (defined over QoS dimensions) to a real number called task utility. System utility, which is the sum of individual task utilities is maximized. Our approach relies on multicriteria techniques and is somewhat simpler, because we ask the user for less information by demanding priorities instead of utility functions.

6.2. Service selection within service composition

Most approaches to service composition incorporate some solution to the service selection problem. Service composition consists of the identification of several services, which together can perform some functionality that none of the individual services can perform alone. In [35, 36] service composition and therefore selection are guided by users' utility functions over QoS dimensions. To select the optimal service candidate, a simple additive weighting technique is used to associate a weight to each criterion. Jaeger, Mühl and Golze [9] highlight similarities to other combinatorial problems as the knapsack problem and the resource constraint project scheduling problem. They discuss about possible heuristics to solve the composition problem and evaluate their efficiencies. A score is assigned to each QoS criterion with a simple additive weighting technique to express its respective importance. Approaches relying on the simple additive weighting technique do not account requester relative preference like the revised Simos procedure used in our approach. In [7] a scalable QoS-aware service aggregation model composing service on-demand while satisfying the user's quality requirements is proposed. It selects services dynamically based on composite and distributed performance information. Each quality criterion is associated to a weight. Grønmo and Jaeger proposes in [6] a model-driven methodology for building web services compositions that are QoS optimized. They present a control flow pattern approach that optimizes the QoS values of the composition given user-defined requirements and preferences. In [30] an approach to e-business process negotiation using a generic iterative bargaining protocol and multi-criteria decision support is proposed. Alternative bids are evaluated with the help of the analytic hierarchy process (AHP) and absolute priorities weighting are used to express preferences over QoS attributes. The AHP as a compensatory approach does not reflect the intrinsic relative importance of the QoS property as good as the revised Simos procedure.

7. Conclusions

Enabling autonomic computing through service-oriented systems requires means for appropriately selecting at runtime among potentially many services that can perform the same tasks. It is otherwise impossible to ensure that only those services are selected that are capable to satisfy requests and honor service level agreements to the most desirable extent. One of the key sources of difficulty in enabling efficient service selection lies in dealing with conflicting QoS expectations of the stakeholders.

We introduce an approach for dealing with trade offs between QoS criteria during runtime service selection. Our approach is designed on the realistic premise that services' performance over QoS criteria is changing, so that observed values need to be used in decision-making, instead of assuming advertised QoS values are maintained indefinetly. More precisely, we introduce a model to define quality attributes and their respective characteristics. Because such properties cannot be simultaneously optimized, we use the notion of priority and integrate it in the model. We show how to express priorities between QoS with the help of weighting factors. In order to establish a ranking over alternative services, we use outranking methods. Such methods generate a rating while considering relative importance (weights) of the observed QoS dimensions. Our approach is generic, that is, can be integrated with available QoS-aware service selection and composition algorithms. To enable this integration, we show throughout the paper how our approach allows the definition of a global priority constraint to be subsequently used as an ordinary constraint in an optimization problem (i.e., the service selection problem). The constraint ensures that only part of the services be considered during selection and composition, rejecting thus automatically the noncompetitive services. The definition of quality characteristics and their respective priorities allows to perform an autonomic selection of "best" available services.

In contrast to most comparable efforts, our QoS model provides a framework applicable to various quality dimensions and is therefore not limited to a set of predefined QoS dimensions. Weights are determined by an algorithm considering user priorities and not predetermined rules. Outranking methods allow us to rely on relative importance of QoS dimensions. Moreover, the global priority constraint that limits the set of relevant services at selection is built from observed and not advertised values. It is therefore more realistic, being appropriate for systems in which advertised QoS levels cannot be guaranteed at all times. Future work will focus on facilitating, through appropriate tools, the the identification and specification of priorities.

References

[1] J. Brans and P. Vincke. A preference ranking organization method. *Management Science*, 31(6):647–656, 1985.

[2] F. Casati, M. Castellanos, U. Dayal, and M.-C. Shan. Probabilistic, context-sensitive, and goal-oriented service. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 316–321, New York, NY, USA, 2004. ACM Press.

[3] A. D'Ambrogio. A model-driven wsdl extension for describing the qos ofweb services. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 789–796, Washington, DC, USA, 2006. IEEE Computer Society.

[4] J. Figueira and B. Roy. Determining the weights of criteria in the electre type methods with a revised simos' procedure. *European Journal of Operational Research*, 127(2):317–326, June 2002.

[5] S. Ghosh, J. Hansen, R. Rajkumar, and J. Lehoczky. Adaptive qos optimizations for radar tracking. In *RTCSA'04: Proceedings of the 10th International Conference on Real-Time and Embedded Computing Systems*, 2004.

[6] R. Grønmo and M. C. Jaeger. Model-Driven Methodology for Building QoS-Optimised Web Service Compositions. In L. Kutvonen and N. Alonistioti, editors, *Distributed Applications and Interoperable Systems: 5th IFIP WG 6.1 International Conference (DAIS'05)*, volume 3543 of *LNCS*, pages 68–82, Athens, Greece, May 2005. Springer Verlag.

[7] X. Gu and K. Nahrstedt. A scalable qos-aware service aggregation model for peer-to-peer computing grids. In *In Proceedings of the IEEE HPDC-11, Edinburgh, Scotland*, 2002.

[8] J. Hokkanen and P. Salminen. The choice of a solid waste management system by using the electre iii decision-aid method. In M. Paruccini, editor, *Applying multiple criteria aid for decision to environmental management*, volume 3 of *Environmental Management*, pages 111–153. Kluwer Academic Publishers, Dordrecht, 1994.

[9] M. C. Jaeger, G. Mühl, and S. Golze. Qos-aware composition of web services: An evaluation of selection algorithms. In *OTM Conferences (1)*, pages 646–661, 2005.

[10] I. J. Jureta, S. Faulkner, Y. Achbany, and M. Saerens. Dynamic task allocation wihin an open service-oriented mas architecture. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agents Systems (AAMAS'07)*, 2007.

[11] I. J. Jureta, S. Faulkner, Y. Achbany, and M. Saerens. Dynamic web service composition within a service-oriented architecture. In *Proceedings of the International Conference on Web Services (ICWS'07)*, 2007.

[12] I. J. Jureta, C. Herssens, and S. Faulkner. A comprehensive quality model for service-oriented systems. In *Software Quality Journal. Accepted for publication (available online at: http://www.jureta.net/papers/QVDPdraft.pdf).*

[13] A. Keller and H. Ludwig. The wsla framework: Specifying and monitoring service level agreements for web services. *J. Netw. Syst. Manage.*, 11(1):57–81, 2003.

[14] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[15] Y. Liu, A. H. Ngu, and L. Z. Zeng. Qos computation and policing in dynamic web service selection. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73, New York, NY, USA, 2004. ACM Press.

[16] M. E. Maximilien and M. P. Singh. Toward autonomic web services trust and selection. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 212–221, New York, NY, USA, 2004. ACM Press.

[17] M. E. Maximilien and M. P. Singh. Multiagent system for dynamic web services selection. In *Proceedings of the International conference on Autonomous Agents and Multi-Agent Systems*, 2005.

[18] S. McIlraith and T. C. Son. Adapting golog for composition of semantic web services. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR'02)*, 2002.

[19] S. A. Mcilraith and D. L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18(1):90–93, 2003.

[20] D. A. Menascé. Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[21] V. Mousseau. Eliciting information concerning the relative importance of criteria. In P. Pardalos, Y. Siskos, and C. Zopounidis, editors, *Advances in Multicriteria Analysis*, pages 17–43. Kluwer Academic, Dordrecht, 1995.

[22] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *Proceedings of the International Conference on the World Wide Web (WWW 2002)*, 2002.

[23] OMG. Uml profile for modeling qos and fault tolerance characteristics and mechanisms. Technical report, Object Management Group, 2006.

[24] J. O'Sullivan, D. Edmond, and A. T. Hofstede. What's in a service? towards accurate description of non-functional service properties. *Distrib. Parallel Databases*, 12(2-3):117–133, 2002.

[25] A. Padovitz, S. Krishnaswamy, and S. W. Loke. Towards efficient selection of web services. In *Proceedings of the Second International Joint Conferences on Autonomous Agents and Multi-agent Systems*, 2003.

[26] M. P. Papazoglou and D. Georgakopoulos. Introduction. *Commun. ACM*, 46(10):24–28, 2003.

[27] M. Pistore, P. Traverso, and P. Bertoli. Automated composition of web services by planning in asynchronous domains. In S. Biundo, K. L. Myers, and K. Rajan, editors, *ICAPS*, pages 2–11. AAAI, 2005.

[28] M. Pistore, P. Traverso, P. Bertoli, and A. Marconi. Automated synthesis of composite bpel4ws web services. In *ICWS*, pages 293–301. IEEE Computer Society, 2005.

[29] T. Saaty. *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. McGraw-Hill, New york, 1980.

[30] S. E. Shaikh and N. Mehandjiev. Multi-attribute negotiation in e-business process composition. In *WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04)*, pages 141–146, Washington, DC, USA, 2004. IEEE Computer Society.

[31] J. Simos. *Gestion des Déchets Solides Urbains Genevois : Les Faits, le Traitement, l'Analyse*. Presses Polytechniques et Universitaires Romandes, Lausanne, 1990.

[32] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller. A concept for qos integration in web services. In *Proceedings of the 1st Web Services Quality Workshop (WQW2003)*, 2003.

[33] H. Tong and S. Zhang. A fuzzy multi-attribute decision making algorithm for web services selection based on qos. In *IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*, volume 0, pages 51–57, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[34] P. Vincke. *Multicriteria Decision-Aid*. J. Wiley, New York, 1992.

[35] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 411–421, New York, NY, USA, 2003. ACM Press.

[36] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.

[37] C. Zhou, L.-T. Chia, and B.-S. Lee. Daml-qos ontology for web services. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 472, Washington, DC, USA, 2004. IEEE Computer Society.