A QoS-Aware Selection Model for Semantic Web Services

Xia Wang[1], Tomas Vitvar[1], Mick Kerrigan[2], and Ioan Toma[2]

[1] Digital Enterprise Research Institute(DERI)
IDA Business Park, Lower Dangan Galway, Ireland
[2] Digital Enterprise Research Institute (DERI),
Leopold-Franzens Universität Innsbruck, Austria
{xia.wang, tomas.vitvar, michael.kerrigan, ioan.toma}@deri.org

Abstract. Automating Service Oriented Architectures by augmenting them with semantics will form the basis of the next generation of computing. Selection of service still is an important challenge, especially, when a set of services fulfilling user's capabilities requirements have been discovered, among these services which one will be eventually invoked by user is very critical, generally depending on a combined evaluation of qualities of services (Qos). This paper proposes a QoS-based selection of services. Initially we specify a QoS ontology and its vocabulary using the Web Services Modeling Ontology (WSMO) for annotating service descriptions with QoS data. We continue by defining quality attributes and their respective measurements along with a QoS selection model. Finally, we present a fair and dynamic selection mechanism, using an optimum normalization algorithm.

1 Introduction

Web services with well-defined semantics, called semantic Web services (SWS), provide interoperability between Web services by describing their own capabilities in a computer-interpretable way [10, 11]. The greatest advantage of SWS is that they enable machines to automatically perform complex tasks by manipulating a series of heterogeneous Web services based on semantics. Most aspects of SWS, such as automatic discovery, selection, composition, invocation, or monitoring of services are tightly related to the quality of these services (Qos). QoS as part of the service description is an especially important factor for service selection [5] and composition [14].

In order to discover services, a service requester provides some requirements on the capability of a requested service. Furthermore, many service providers publish their services by advertising the service capabilities. Hence a service discovery engine can be used to match requirements of a user against advertised capabilities of service providers. In such a case that several similar services are yielded by the discovery process, which has carried out the matchmaking of the non-functional and functional properties of services. Of these similar services, the one which will be finally invoked by the user depends mostly on the qualities of services.

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 390–401, 2006.

In the literature, this issue has not been thoroughly addressed, due to the complexity of QoS metrics. Sometimes, the quality of a service is dynamic or unpredictable. Moreover, most of the current work focuses on the definition of QoS ontology, vocabulary or measurements and to a lesser extent on an uniform evaluation of qualities. In our former work, we defined a selection model for semantic services [12][1], in which we specified details of quality-based selection algorithm. This paper will go on to elaborate the synthetical evaluation of the multiple and diverse qualities of services for selection of service.

The Web Service Modeling Ontology (WSMO) [13] is a conceptual model for describing Web services semantically, and defines the four main aspects of semantic Web service, namely Ontologies, Web services, Goals and Mediators. With respect to WSMO, only a small amount of work has been carried out on the selection of services, mainly in [3], which introduces a number of generic selection mechanisms to be used conjunction with WSMO. In this paper, we use the WSMO model and features to describe a QoS model, specific quality metrics, value attributes, and their respective measurements. Furthermore, we propose an algorithm to normalize different quality attributes, providing a dynamic and fair evaluation of services. This is done by considering users' quality requirements together with a set of quality advertisements provided by a service provider. Then we synthetically evaluate all of the metrics closeness in quality attributes by normalization. A weight matrix is applied to obtain the final evaluation.

The paper is structured as follows, Section 2 provides an overview on the current related work. In Section 3, a QoS ontology language designed for the needs of Web services is defined in the context of WSMO, and a QoS model for service selection is presented. Our QoS-based service selection algorithm is evaluated in Section 4. In Section 5 experimental results are presented to show the validity of the algorithm.

2 Related Work

Most of the related work in using QoS for service selection focuses on the development of QoS ontology languages and vocabularies, as well as on the identification of various QoS metrics and their measurements with respect to semantic services. For example, [9] and [4] emphasized the definition of QoS aspects and metrics. In [9], all of the possible quality requirements were enumerated and organized into several categories, including runtime-related, transaction support related, configuration management and cost-related QoS, and security-related QoS. Also, they shortly present their definitions or possible determinants. Unfortunately, they failed to present quantifiable measurements.

In [8] and [2], the authors focused on the creation of QoS ontology models, which proposed QoS ontology frameworks aiming to formally describe arbitrary QoS parameters. From their on-going work, we know that they did not consider, yet, QoS-based service matching. Additionally, the work [5], [6], and [7]

[1] This research was supported by FernUniversitaet, in Hagen and by DAAD, the German Academic Exchange Service.

tries to attempt to conduct a proper evaluation and proposes QoS-based service selection, despite the authors failing to present a fair and effective evaluation algorithm.

Especially, the work was presented in [5], which is also similar to ours. There are, however, some differences to our approach: 1) The measurement of linguistic-based qualities was not considered; 2) The algorithm uses average ranking, neglecting nuances in different quality properties; 3) A possible maximum value is used to normalize the QoS matrix, although such kind of value is worth deliberating; 4) Upon analyzing the experimental data, after normalization, the final result looks as $G' = (\{0.769, 1.429, 1.334, 1.111\}, \{0.946, 0.571, 0.666, 0.889\})$. For their way of normalisation, it is hard to make a fair evaluation of all qualities, because the metrics do not have the same range. One quality attribute even has a higher weight, while its real impact is decreased by its smaller value. Therefore, our approach is to normalize each quality metric into values between 0 and 1 by specifically defined measurements, which are fair to each quality metric. That means, we propose a different normalization algorithm.

Additionally, [17] focused on augmenting QoS classes and properties to extend the DAML-S [1] profiles. [16] defined a QoS ontology for DAML+OIL using description logic notions to express different QoS templates. [9] incorporated QoS into UDDI and SOAP messages [4] to improve the service discovery process. This paper however emphasizes the extension of WSMO with a QoS ontology class.

3 QoS Ontology Language and Vocabulary in WSMO

The Web Service Modeling Ontology (WSMO) [13] is a conceptual model for describing various aspects related to semantic Web services. WSMO is made up of four top level elements, namely ontologies, web services, goals and mediators. Briefly, *ontologies* provide the terminology and formal semantics for the other elements of WSMO. *Web services* define a semantic description of services including their functional and non-functional properties. *Goals* specify the requesters requirements for a Web Service. And *mediators* resolve the heterogeneity problem by implementing ooMediators (between ontologies), ggMediators (between goals), wgMediators (between web services and goals), and wwMediators (between services).

In WSMO, quality aspects are part of the non-functional information of a Web service description and are simply defined as: *Accuracy*, *Availability*, *Financial*, *Network-related QoS*, *Performance*, *Reliability*, *Robustness*, *Scalability*, *Transactional* and *Trust*. Such kinds of QoS definition are neither expressive nor flexible enough for QoS attributes. Therefore in this paper, for the purpose of selecting services, we introduce a new class, QoS concept classes, that refines the non-functional properties class in WSMO. Furthermore, we define a QoS model following the same syntax to extend the WSMO model. The defined QoS model may be referred to by the *web service* and *goal* entities, and quality factors can adequately be considered during the process of service selection.

We will specify a QoS upper ontology named WSMO-QoS. It is a complementary ontology that provides detailed quality aspects about services. Developers benefit from WSMO-QoS for QoS-based matchmaking and QoS measurement.

3.1 QoS Ontology and Vocabulary

Based on [2, 6, 8], we define a new class *QoS* (Table. 1) which is a subclass of *nonFunctionalProperties* class already defined in WSMO. Class *QoS* can be attached to class *webService* or *Goal*. Please notice that the current WSMO conceptual model remains unchanged, we however simply refine the class *nonFunctionalProperties*.

Table 1. QoS Ontology in WSMO

Class QoS **sub-Class** nonFunctionalProperties
hasMetricName **type** string
hasValueType **type** valueType
hasMetricValue **type** value
hasMeasurementUnit **type** Unit
hasValueDefinition **type** logicalExpression
multiplicity = single-valued
isDynamic **type** boolean
isOptional **type** boolean
hasTendency **type** {small, large, given}
isGroup **type** boolean
hasWeight **type** string

Each QoS metric is generally described by *MetricName*, *ValueType*, *Value* (given or calculated at service run-time), *MeasurementUnits* (e.g. $, millisecond), *ValueDefinition* (how to calculate the value of this metric), and *Dynamic/Static*. For the purpose of QoS-based selection, there are four additional features defined, namely: *isOptional*, *hasTendency*, *isGroup*, and *hasWeight*. The following is an simple interpretation of every property in Table. 1:

- Types of the parameter *valueType* may be *linguistic*, *numeric* (int, float, long), *boolean* (0/1, True/False) or other. Therefore, there will be different forms of preprocessing according to the different value types.
- The property $MetricValue$ defines a metric's values which are either real ones or a string such as $'calculate'$. If $MetricValue =' calculate'$, then this attribute should refer to its $valueDefinition$ for a dynamic value calculation.
- The property *MeasurementUnit* specifics the concrete unit of every quality metric, with possible types such as $Unit = \{\$, millisecond, percentage, kpbs, times, ...\}$. In addition, class *Unit* has a conversion function between different measurement units, e.g., to transform second to millisecond.

- Parameter $hasValueDefinition$ is either a logical expression defined as in [13] or the string $'NULL'$. If $hasValueDefinition =' NULL'$, then this value definition cannot explicitly extracted from the context of service description, but must dynamically be invoked from its service provider. In this case this quality attribute must be dynamic, that is $isDynamic = True$.
- Through property *isDynamic*, the nature of a quality is defined as static or dynamic. For a static quality, its values are given by a priori, and can be directly used during the selection process. If $isDynamic = True$, this quality metric must be dynamically invoked and obtained from its service provider, and its values must be calculated at run-time.
- If $isOption = 0$, this attribute, assumed to be noted as q_k, is necessary, such that $q_k \in Q_N$, where Q_N is the necessary quality set. This property is described in Subsection 3.2.
- *hasTendency* is an object property representing the expected tendency of the value from the user's perspective. For example, the price of a service is expected to be as low as possible, so that its $hasTendency =' low/small'$. On the contrary, the quality of *security* of a service should be as high as possible, i.e., $hasTendency =' high/large'$. When $hasTendency =' given'$, the user expects the value of this quality to be as close the given value as possible. Also, in a quality inquiry, $hasTendency = \{low/small, high/large, given\}$ denotes, respectively, that $\{\geq, \leq, =\}$ for its $MetricValue$.
- *isGroup* indicates if this quality attribute is defined by a group of other qualities or not. For example, *security* is composed of *nonRepudiatior*, *DataEncryption*, *Authorisation*, *Authentication*, *Auditability*, and *Confidentiality* [2]. Hence, $isGroup = True$ means that in the preprocessing stage, the group value must be calculated first.
- Finally, $hasWeight$ is a value denoting the weightiness of the property, especially when synthetically measuring several metrics. In this context we define the weight value either ranges in $[0, 10]$ or 'NULL', different end users have different weight values for their service requirements. Note, in this paper, this property is used only by a WSMO *goal*, which describes user's desire; In the description of a WSMO *web service*, its value is 'NULL'.

During the selection process, when a QoS profile is parsed, in order to obtain a metric's value for which $hasMetricValue =' calculate'$ holds, its $hasValueDefinition$ property must be checked to determine how to calculate it. If $hasValueDefinition =' NULL'$ and $isDynamic = 1$, then the invocation function is to inquire the real-time value, otherwise an error is encountered. If $isDynamic = 0$, its corresponding $hasMetricValue$ is an existing value, or again an error occurs.

In [15,4,9], all of the possible QoS requirements for Web services were defined, mainly including: performance, reliability, scalability, capacity, robustness, exception handling, accuracy, integrity, accessibility, availability, inter-operability, security and network-related QoS requirements. Fig. 1 gives a simple view on QoS vocabulary, which consists of many general QoS attributes and the scalable domain-specific QoS subset used; for example, to define the hotel category for a hotel service. The definition and the discussion of concrete measurement of qualities is out of this paper's scope.

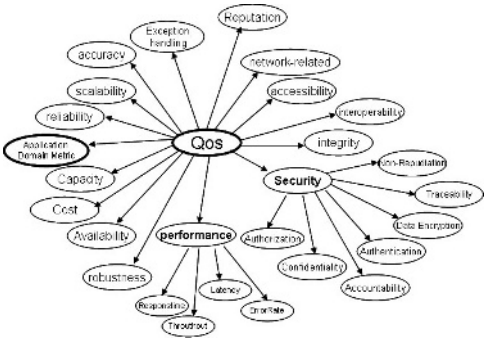

Fig. 1. QoS ontology and vocabulary

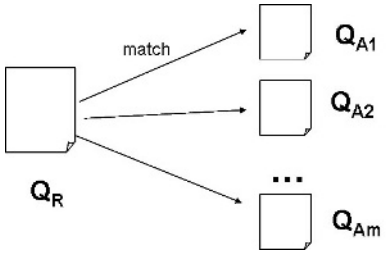

Fig. 2. QoS-based selection of services

3.2 QoS Selection Model

The scenario of QoS-based service selection is described as follows. The user provides his requirements (including non-functional, functional, and quality properties) for the expected service, which are formed into a requirement profile, noted as $s_R = (NF_R, F_R, Q_R, C_R)$, where the denotations are the identifers of Non-Functionality, Functionality, Quality and Cost (the details of such selection model can be found in [12]). On the other side, there can be thousands of available services published in either a service repository or a kind of peer-to-peer service environment. The advertisement of a service s is denoted as $s_A = (NF_A, F_A, Q_A, C_A)$, similarly.

The first filter of service selection matches s_R with any available s_A on the basis of non-functional-NF (bascially only the service name and service category) and functional-F (including inputs, outputs, preconditions and effects) features of services. We assume that m similar services are yielded, namely, $S = \{s_1, s_2, ..., s_m\},\ m \in \mathcal{N}$.

The second filter synthetically considers all quality features to select the service among S satisfying the user's requirements best. This matchmaking takes place between the pair of the QoS requirements Q_R and a quality profile Q_A of a candidates service $s_A \in S$, as illustrated in Fig. 2.

For the purpose of matching, a QoS selection model is defined, in which metrics are defined both from the perspectives of users and providers of web services. We assume that $Q = \{q_1, q_2, ..., q_i\},\ i \in \mathcal{N}$, and Q_I denotes the quality set. Thus,

- Q_N is the necessary quality set for each service defaulted by machine, and $Q_N \subseteq Q_I$;
- Q_O is the optional quality set of the service defined as $Q_O = Q_I \setminus Q_N$; and
- Q_D is the default quality set of the service. When user does not explicitly give any quality requirements, i.e., when $Q_R = \emptyset$ and $Q_D \subseteq Q_I$, then Q_D will be taken as Q_R, i.e., $Q_R = Q_D$, where Q_R are the user's quality requirements. Generally, $Q_N \subseteq Q_D$.

There are two reasons for distinguishing between different QoS sets. One is to free the customer from multifarious definitions of his quality requirements, which sometimes need professional knowledge. For example, a customer cannot understand the meaning of *availability* of a service, but he apparently has a requirement for it. So, the customer may only provide qualities based on his personal opinions, whereas the complementary part is left to be defined in the default quality set. The other reason is for the simple, high effective QoS-based approach for service selection.

Basing on the above analysis, there are three kinds selection modes with respect to Q_R:

- Default mode. When $Q_R \neq \emptyset$, Q_R is redefined as union of the original user requirements and the default ones about service performance, as $Q_R := Q_R \cup Q_N$;
- Totally based on the user's requirements, and $Q_R \neq \emptyset$;
- Totally based on default definitions, if $Q_R = \emptyset$.

Further, for purposes of efficient and flexible service selection and from the user's perspective, in our model only several qualities are defined in the necessary set, viz., Q_N={*cost, responseTime, reliability, accurary, security, reputation*}, and similarly Q_D={*cost, responseTime, reliability, accuracy, security, reputation, executionTime, exceptionHandling*}. Of course, the definitions are extendable and changeable for specific application system.

There are many approaches to collect values of quality metrics:

- Directly from the service descriptions, e.g., sometimes the price of invoking a service is given a priori.
- Simple calculation of a quality value based on the defining expression in the service description.
- Collection through active monitoring, e.g., execution duration defined in [5].
- Dynamical inquiry from the current server.
- Periodical update of quality values for statistical purposes in a log.
- Obtaining the customers' feedbacks on quality characteristics, e.g., *Reputation* of a service [5] .

Not only are the collection of quality requirements dynamic, unpredictable, and even difficult during run-time, but the value characteristics of quality metric can be concluded approximately as:

- Numerical metric, denoted by a number but with different value ranges.
- Ordinal and linguistic-based metric, denoted by a term from an ordered finite collection of terms, e.g., the reputation of a service may be evaluated by $\{Low, veryLow, Medium, veryHigh, High\}$.
- Regional metric, denoted by a numerical region $[min, max]$.
- Graded metric, e.g., rank of a hotel service in $\{1, 2, 3, 4, 5\}$.
- Boolean value numeric or enumerative scales.

It is worth noting that this QoS model is easy to extend or customize. The user may customize his/her Q_N, Q_D, Q_I at will. The detailed definition of all quality attributes is out of this paper's scope. Instead, we focus mainly on the QoS foundation of the selection model, and the combined evaluation of the quality attributes.

4 Selection Algorithm

QoS-based selection of services is very complex, not only due to the diversity of multifarious quality metrics with different value types, value range, and measurements, but also since an effective algorithm, which evaluates all metrics in combination, is missing.

We assume that $Q_R = \{r_1, r_2, ..., r_k\}$ expresses the profile of a user's quality requirements, which includes k quality metrics. Similarly, the quality profile of m candidate services in set S is denoted as $Q_S = \{Q_{A_1}, Q_{A_2}, ..., Q_{A_m}\}$, where $Q_{A_i} = \{q_{i1}, q_{i2}, ..., q_{ij}\},\ i, j \in \mathcal{N}$. It defines that the advertisement of service S_i has j quality metrics provided.

It is well-known that there are two cases during the matchmaking,

- $Q_R = \emptyset$, then $Q_R := Q_D$;
- $Q_R \neq \emptyset$, then $Q_R := Q_R \cup Q_N$. The Q_R is matched with each $Q_{Ai},\ i \in \mathcal{N}$.

It is quite obvious that it is rather unlikely that any Q_R or Q_{A_i} will have the same number of quality metrics. So, in the first preprocessing step, we take Q_R as benchmark for alignment with every Q_{A_i}. This process includes:

1. To re-arrange the metrics of Q_{Ai} in the same order.
2. If Q_{A_i} is lacking a quality, then one can add a metric and set its value to 0.
3. To tailor the qualities which are not listed in Q_R.

Therefore, the matrix of QoS for service matchmaking $M_Q = \{Q_R, Q_{A_1}, Q_{A_2}, ..., Q_{A_m}\}$ looks like:

$$M_Q = \begin{pmatrix} r_1 & r_2 & r_3 & \dots & r_k \\ q_{11} & q_{12} & q_{13} & \cdots & q_{1k} \\ q_{21} & q_{22} & q_{23} & \cdots & q_{2k} \\ \dots & \dots & \dots & \dots & \dots \\ q_{m1} & q_{m2} & q_{m3} & \cdots & q_{mk} \end{pmatrix}_{(m+1)\times k}$$

Here, M_Q is a $(m+1) \times k$ matrix, with the quality requirements Q_R in the first row, and the quality information of candidates services in the other rows. Each column contains values of the same quality property. For uniformity, matrix M_Q has to be normalized with the objective to map all real values to a relatively small range, i.e., the elements of the final matrix are real numbers in the closed interval $[0, 1]$. The main idea of the algorithm is to scale the value ranges with the

maximum and minimum values of each quality metric for thousands of current candidate services. Accordingly, the maximum and minimum values are mapped to the uniform values 1 and 0, respectively, depending totally on their definition of *hasTendency*.

For instance, a user searches a flight constraining the ticket price to be below \$300, and three service providers ask for \$250, \$280, and \$260, respectively. In this case the minimum and maximum are \$250 and \$280. Then, the calculation of relative closeness for this quality metric reads as $(1-\frac{250-250}{280-250}) = 1$, $(1-\frac{280-250}{280-250}) = 0$, and $(1-\frac{260-250}{280-250}) = 0.667$.

The second preprocessing step is uniformity analysis. We distinguish different quality metrics with their value features. In our QoS model, we take the information of *hasTendency* as a quality metric $r_i, i \in k$:

1. if $hasTendency =' given'$, then we calculate the ratio by

$$q'_{ij} = \begin{cases} 1 - \frac{q_{max} - q_{ij}}{q_{max} - q_{min}} & \text{if } r_j \geq q_{max} \\ \frac{q_{ij} - q_{min}}{q_{max} - q_{min}} & \text{if } r_j \leq q_{min} \\ 1 - (|\frac{|q_{ij} - r_j| - m}{n - m}|) & \text{if } r_j \in (q_{min}, q_{max}) \end{cases} \tag{1}$$

2. if $hasTendency =' small/low'$, then the ratio is calculated by

$$q'_{ij} = (1 - \frac{q_{ij} - q_{min}}{q_{max} - q_{min}}) \tag{2}$$

3. if $hasTendency =' large/high'$, then the ratio is calculated by

$$q'_{ij} = (1 - \frac{q_{max} - q_{ij}}{q_{max} - q_{min}}) \tag{3}$$

where $q_{max} = \max\{q_{ij}\}$, $q_{min} = \min\{q_{ij}\}$, $n = \max\{|q_{ij} - r_{ij}|\}$, and $m = \min\{|q_{ij} - r_{ij}|\}$, $i \in k, j \in m$. In Fig. 3, three cases of matchmaking are shown, and the area from the left to the right of the scale line corresponds to the growing values, whose tendency is *small/low*, *given*, and *large/high*, respectively. Also the value of $r_j,\ j \in k$ is scattered either among $q_{ij},\ i \in m$ or the right side or the left side of the candidates values. Formula 1-3. present their algorithms.

By taking the Formula 1. as an example, it describes the case that a user requires the value of a quality to be as close to his given value as possible. We assume r_j with its value as u_j and the other quality $\{q_a, q_b, ..., q_h\}$ with their value as $\{v_a, v_b, ..., v_h\}$. There are also three cases in Formula 1. First, when $u_j \geq q_{max}$, just as the candidate set is $\{q_a, q_b, q_d, q_d\}$, then by Formula 1 we know q_d gets the best ratio as 1. A similar situation occurs when $u_j \leq q_{min}$. When r_j scatters in $\{q_c, q_d, q_e, q_f\}$, the range of scale should be first defined by $(n - m)$, then ratios are calculated following the third case of Formula 1.

The weighted value for each quality metric is defined in the parameter of *hasWeight*. These are brought into the form of a diagonal matrix as $W = \{w_1, w_1, ..., , w_k\}$. Here, we assume that $\sum_{i=1}^{n} w_i = 10$ (which is not defined as 1,

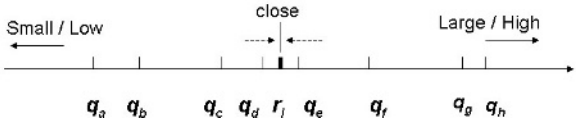

Fig. 3. Quality Measurement

for the reason of magnifying the effect of experiments). Then, W is applied to matrix M_Q yielding

$$M_{Q'} = M_Q \times W = \sum_{i=1}^{m} (q'_{ij} \times w_i) \tag{4}$$

Finally, we can calculate the evaluation result for each quality metric by summing the values of each row. These abstract values are taken as a relative evaluation of each service's QoS.

5 Experiments

For reasons of comparison and simplification we borrowed test data from [5]. In their experiments, they implemented a hypothetical phone service (UPS) registry, which provides various phone services such as long distance, local, wireless, and broadband. They simulated 600 users to collect the experimental data. Especially, two phone services' test data are presented with seven quality criteria, including *Price*, *Transaction*, *Time Out*, *Compensation Rate*, *Penalty Rate*, *Execution Duration*, and *Reputation*. Their corresponding value types are \$, 0/1, microsecond, percent, percent, microsecond, and rank value in $[0, 5]$.

In order to be applied into our selection mode, we assume a requirement of a service customer and another two services for testing, then the M_Q is as Table.2.

Table 2. Experiment Data

Data	Pri	Trans	TimeOut	ComRat	PenRat	Execu	Repu
R	30	1	80	0.4	0.8	120	4.0
ABC	25	1	60	0.5	0.5	100	2.0
BTT	40	1	200	0.8	0.1	40	2.5
A_1	28	1	140	0.2	0.8	200	3.0
A_2	55	1	180	0.6	0.4	170	4.0

The first row is the supposed Q_R, the next two rows are taken from [5], and the last two ones are also hypothetical candidates services. From the definitions of each quality criterion of that example, we know that *Price* and *Execution Duration* are expected to be smaller, *Compensation Rate*, *Penalty Rate*, and *Reputation* are to be bigger, and *Time Out* is required to be as close as possible.

The result of normalization carried out by our algorithm for the four candidate services referring to Q_R is:

$$Q' = \begin{pmatrix} 1 & 1 & 0.870 & 0.500 & 0.571 & 0.625 & 0 \\ 0.500 & 1 & 1 & 1 & 0 & 1 & 0.250 \\ 0.900 & 1 & 0.522 & 0 & 1 & 0 & 0.500 \\ 0 & 1 & 0 & 0.667 & 0.429 & 0.188 & 1 \end{pmatrix}$$

Assuming $W = \{4, 0, 0, 2, 1, 1, 2\}$, we apply Formula 4. to obtain a quality evaluation set, named $Q'' = \{6.196, 5.500, 5.600, 3.951\}$. That is, in case of putting a high weight on price, service s_1 is the best choice, the order of the results is in line with human intuition, see Fig. 4, and the result is consistent with [5], too.

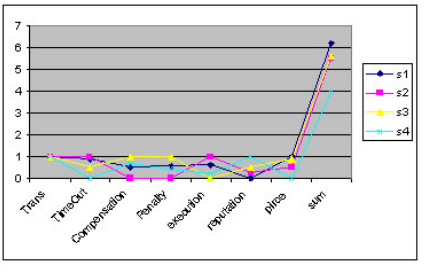

Fig. 4. Combined evaluation of qualities

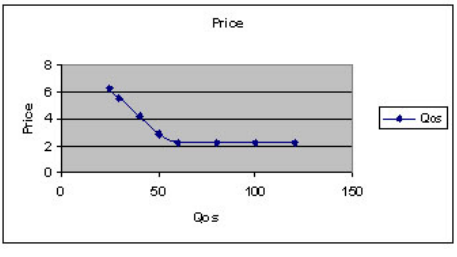

Fig. 5. Combined evaluation of qualities

Here a short discussion is presented. Our Qos-based model is dynamic and real-time, which is fully adapted to the current distributed network environment, and which is kept a well relativity and up-to-date, it is also fair on this point. Since it is always basing on the current available services to compare their current integrative capability. If services are added or deleted, the evaluation should be updated.

Also, in a certain relatively stable service environment, a service provider may consider to change one of its property, it is easy to forecast its constraint for value. For instance, we take the service s_1 (the service ABC in Table. II) as an example to analysis the effect of the price on its QoS. From Fig. 5., we knew that if its price were to go beyond the current maximum, it will lose its competition on price and keep an invariable QoS value.

6 Conclusion

This paper proposed a QoS-based approach for web service selection, by presenting a fair and simple algorithm for evaluating multiple quality metrics in combination. First, we specified a QoS ontology and its vocabulary in order

to augment the QoS information in WSMO. Furthermore, various quality attributes, their respective measurements, and a QoS selection model were defined in detail. Finally, a fair and dynamic selection mechanism was presented, which uses a normalization algorithm oriented at optimal value range. This approach was validated by a case study for a kind of phone service.

Acknowledgment. This material is based upon work supported by the Science Foundation Ireland under Grant No. 02/CE1/I131, and the European projects KnowledgeWeb (FP6-507482), and Adaptive Services Grid (FP6-C004617).

References

1. DAML-S Coalition, DAML-S: Web Service Description for the Semnatic Web. In Proc. Internatioal Semantic Web Conference (ISWC02), 2002.
2. D.T. Tsesmetzis, I.G. Roussaki, I.V. Papaioannou and M.E. Anagnostou, QoS awareness support in Web-Service semantics, AICT-ICIW06, 2006, pp.128-128.
3. M. Kerrigan, Web Service Selection Mechanisms in the Web Service Execution Environment (WSMX), In Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC), Apr 2006, Dijon, France.
4. K. Lee, J. Jeon, W. Lee, S. Jeong and S. Park, QoS for Web Services: Requirements and Possible Approaches, W3C Working Group Note 25, 2003.
5. Y. Liu, A.H.H. Ngu and L. Zeng, QoS Computation and Policing in Dynamic Web Service Selection. Proceeding 13th International Conference World Wide Web, 2004.
6. Y. Mou, J. Cao, S.S. Zhang, J.H. Zhang, Interactive Web Service Choice-Making Based on Extended QoS Model, CIT 2005, pp.1130-1134.
7. D.A. Menasce, QoS Issues in Web Services. IEEE Internet Computing, 2002, 6(6).
8. I.V. Papaioannou, D.T. Tsesmetzis, I.G. Roussaki, and E.A. Miltiades, QoS Ontology Language for Web-Services, AINA2006.
9. S.P. Ran, A Model for Web Services Discovery with QoS. SIGecom Exchange, 2003, 4(1):1-10.
10. S. McIlraith, T.C. Son and H. Zeng, Semantic Web Services, IEEE Intelligent Systems, Special Issue on the Semantic Web, 2001, 16(2):46–53.
11. S. McIlraith and D. Martin, Bringing Semantics to Web Services, IEEE Intelligent Systems, 2003, 18(1):90–93.
12. X. Wang, Y. Zhao, B.K. Kraemer and H. Wolfgan, Representation and Discovery of Intelligent E-Services. In: *E-Service Intelligence – Methodolgies, Technologies and Applications*, Lu, J., Ruan,D., and Zhang, G.(Eds.) 2006.
13. D. Roman, H. Lausen, and U. Keller, D2v1.1. Web Service Modeling Ontology (WSMO), WSMO Final Draft 10 February 2005.
14. L.Z. Zeng, B. Benatallah, H.H.Ngu Anne, M. Dumas, J. Kalagnanam and H. Chang, QoS-Aware Middleware for Web Services Composition, IEEE Transaction Software Engineer, 2004, 30(5):311-327.
15. A. Mani and A. Nagarajan, Understanding Quality of Service for Web Services, IBM Developerworks, 2002.
16. C. Zhou, L.T. Chin and B.S. Lee, DAML-QoS Ontology for Web Services. In International Conference on Web Services (ICWS 2004), 2004, pp.472-479.
17. C. Zhou, L.T. Chia and B.S. Lee, Semantics in Service Discovery and QoS Measurement, IT Professional, 2005, 7(2):29–34.