2012 IEEE 19th International Conference on Web Services

User-Centered QoS Computation for Web Service Selection

Chunqi Shi, Donghui Lin, Toru Ishida
Department of Social Informatics, Kyoto University
Yoshida-Honmachi,Sakyo-Ku, Kyoto, 606-8501, Japan
Email: shi@ai.soc.i.kyoto-u.ac.jp, {lindh,ishida}@i.kyoto-u.ac.jp

***Abstract*—QoS computation plays an important role in Web service selection. It involves property value preprocessing aspect, user satisfaction calculation aspect, and aggregation of multiple QoS properties aspect. However, little attention has been paid to users participating in QoS computation. In this paper, we examine QoS computation from the angle of experienced users and novice users. An experienced user is able to be more active in providing configuration information such as expected boundary of a QoS property, distribution function of user satisfaction, and the aggregation weight of each QoS property. While a novice user has limited experience to do this. Based on the study of user-centered factors in QoS computation, we propose a user-centered QoS computation, which provides a new choice of normalization in property value preprocessing aspect, an approach of approximation in user satisfaction calculation aspect, and a weight suggestion way in aggregation of multiple properties aspect. A case study in translation service selection shows that the proposed user-centered QoS calculation is more efficient for novice users than random configuration, and much more efficient for experience users.**

***Keywords*-quality of service; service selection; user-centered;**

I. Introduction

QoS based selection becomes important for users in finding a proper Web service according to non-functional properties, when faced with multiple functionally equivalent Web services. First, it involves matching functionally equivalent candidates according to service description [1], [2]. Second, it involves QoS computation for selecting from among candidates based on QoS properties [3], [4], [5], [6]. Moreover, during the QoS computation, the requirements of users can be submitted, and the satisfaction degree to which each service candidate meets these requirements can be calculated [5], [7], [8], [9].

Current QoS based selection schemes do not pay much attention to user usability. For example, they generally assume that the weights of properties being aggregated are assigned by the user, but novice users are reluctant to provide such weights, because of their lack of experience. Sometime, they have to ask experienced users for hints on weight setting. clearly, the lack of experience with configuration will affect the final selection result. The user may have no idea of the exact meaning of weights, yielding poor configuration. From the perspective of users, current QoS computation does not consider the participation of users in a facile and flexible way.

In service computing, how to promote usability for users has become a topic of increasing interest. Due to the different interests, knowledge for decision, and preferences, the role of users has been paid closer attention to various issues in service computing. For example, with regard to Web service discovery, beyond functional description adaptation, cooperative discovery was designed to increase practicability of matching Web services [10]. Moreover, the request history has been utilized to associate service candidates [11]. Even the discovery process has been studied and redesigned to yield easy-to-use operation [12]. With regard to QoS query formulation, a non-expert view has been taken, and a guide interface, which is a process wizard to guide user to formulate a query, has been designed [3], [13]. With regard to QoS metric definition, an interaction allows users to define personal metrics according to their skill or preference [14]. Even a reputation-based system has been designed according to such QoS metric definition [15].

In this paper, we focus on promoting the usability of QoS computation. Currently, many studies have examined QoS computation [2], [3], [4], [5], [6], [7]. Here, the QoS computation is reexamined from the angle of participating users by drawing a line between *experienced* users and *novice* users. This elucidates many usability problems, which can affect the computation of final results. For example, an experienced user, but not a novice, may have an idea of the boundary range of certain QoS property value. An experienced user may have image of their own satisfaction on certain QoS property values, but a novice user might be less confident of it. Besides, an experienced user might want to use this service for many times, while a novice user might use this service for only a few times, thus it seems unreasonable for a novice user to expend the same effort as an experienced user in mastering QoS calculation, for example, providing the detailed requirement information. However, existent research on QoS computation has paid no attention to such aspects.

Based on the above consideration, we propose user-centered QoS computation. First, we study the user-centered factors in the QoS calculation, from a view of telling apart experienced users and novice users. For each factor, we examine its potential usage and influences. Based on

978-0-7695-4752-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ICWS.2012.18

this examination, we provide choices and suggestion for experience/novice users, which will make the interaction much easier for them. These choices and suggestions are adopted by out user-centered QoS based selection of Web service.

II. QoS Computation

The models of QoS based Web service selection have been intensely researched and discussed. A QoS computation model is designed with mainly two steps: the normalization of QoS property values, and ranking of aggregated utility [2], [4], [6], [16], [17], [18], [19], [20]. In these steps, divergence is seen on whether to adopt filtering [6], different normalization [4], [17], etc. Further, the importance of diverse requirements on QoS properties from different users is discussed and argued such that the satisfaction of the user requirement should be taken into consideration. As a result, QoS requirements are studied and imported in QoS computation [1], [5], [7], [8], [9]. In this step, divergence is seen with regard to requirement matchmaking [5], [8], [9], satisfaction utility [7], etc.

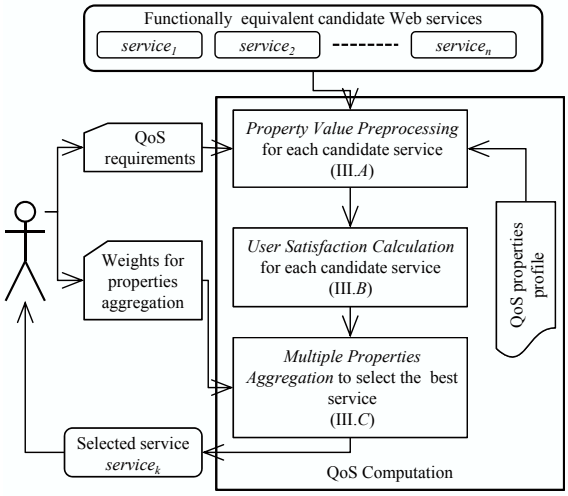


Figure 1. General QoS computation
(more details in Section III.*A*, III.*B*, and III.*C*)

Based on above proposals, Figure 1 depicts the general QoS computation; aspects such as Web service discovery, functional query and matching, QoS monitoring and metric evaluation, etc are ignored. There are three key operations: property value preprocessing (for example, normalization), satisfaction calculation, and multiple properties aggregation. In this general model, users are assumed to provide QoS requirement and weight for aggregation of multiple QoS properties, both of which represent their intent, satisfaction. Indeed, the inner calculation mechanism of each step will affect the final selection results, for example, the $service_k(1 \leq k \leq n)$ (see Figure 1), can be changed when the weights are changed, thus the user received quality will be affected.

III. User-Centered Factors in QoS Computation

We examine the three operations to locate the main user-centered factors. Our strategy is to examine differences in the participation of experienced and novice users in each operation, and then to summarize those differences.

A. Property Value Preprocessing

During this step, the main target is how to preprocess each QoS property value independently from its evaluation metric (for example, the measurement unit). Generally, such calculation is called normalization. Based on existent QoS computation studies, the calculations of property value are categorised into two types: min-max normalization [6], [8], [16], [18], [19], [20], not min-max normalization [4], [17]. Obviously, the former is much more popular, and it is easy to normalize into range $[0, 1]$.

However, there is a deficiency on this min-max type; it requires static boundary of QoS property values. Some property values are available before execution, for example, the *cost*. However, some property values are available only after execution, for example, the *time*. Of course, it is possible to design selection by using history data, i.e. the average response time for each Web service. However, dynamic selecting is important, especially for a domain specific QoS property, for example, *translation quality* of machine translation service. Because it is determined by not only the service, but also the input source. Thus, we can not predict the translation quality by assessing historical data. Besides, when users dynamically select one service from a set, prior expertise in setting the static boundary of property values becomes of little user. For such dynamic selection, it is not easy to predict the exact static boundaries of certain property values.

An example of translation service selection is used to describe the deficiency of min-max normalization (see Table I). Four candidate translation services, *Google*, *J-Service*, *WEB-Transfer*, *Translution*, are available. The properties *cost* and *translation quality* are used for service selection. We dynamically select service results for two source sentences (s_1 and s_2). For sentence s_1, *BLEU* score is used as translation quality, while for sentence s_2 *WER* score is used. Here, *BLEU* and *WER* are two famous automatic machine translation evaluation metrics, and their efficiencies are affected by the feature of input, for example, the length of input sentence. It is a situation that multiple metrics exist for one domain specific property [21].

Normalization is performed according to the min-max equation (1). The QoS property value q_{ij} is the result of evaluating the property p_j of Web service s_i by a mapping metric. l_i is the bottom boundary of the ith property (p_i) values, and its upper boundary is u_i. If a metric is the type

Table I
EXAMPLE OF SELECTING BETWEEN CANDIDATE TRANSLATION SERVICES, GOOGLE(*G*), J-SERVER(*J*), WEB-TRANSFER(*W*), AND TRANSLUTION(*T*), FOR SENTENCE s_1 AND SENTENCE s_2

Input	Property	Metric	Translation Service			
			G	*J*	*W*	*T*
s_1	cost	Dollar	0	2	6	8
	translation quality	BLEU	0.82	0.84	0.86	0.88
s_2	cost	Dollar	0	2	6	8
	translation quality	WER	-0.8	-0.6	-0.4	-0.2

Table II
MIN-MAX NORMALIZATION FOR EACH CANDIDATE TRANSLATION SERVICE, GOOGLE(*G*), J-SERVER(*J*), WEB-TRANSFER(*W*), AND TRANSLUTION(*T*), FOR SENTENCE s_1 AND SENTENCE s_2

Input	Property	Translation Service			
		G	*J*	*W*	*T*
s_1	cost	1	0.75	0.25	0
	translation quality	0	0.33	0.66	1
s_2	cost	1	0.75	0.25	0
	translation quality	0	0.33	0.66	1

of gaining, it is called *positive*. Otherwise, the metric is type of the paying, and it is called *negative*.

$$q'_{ij} = \begin{cases} (q_{ij} - l_j)/(u_j - l_j), & \text{if metric is positive} \\ (u_j - q_{ij})/(u_j - l_j), & \text{if metric is negative} \end{cases} \quad (1)$$

If the maximum and minimum values (dynamically calculated) as the boundary, then $l_j = \min_i\{q_{ij}\}$, and $u_j = \max_i\{q_{ij}\}$, making it easy to normalize the property values (see Table II).

The normalized results for sentence s_1 and s_2 are exactly the same, which seems unreasonable:

- For s_1, there is no big difference between *BLEU* values of *Google* (0.82) and *Translution* (0.88). Considering the big difference between *cost* value of *Google* (0$) and *Translution* (8$), this similarity of *translation quality* suggests that we should emphasize *cost*.
- For s_2, by selecting *Google* (-0.2), its *WER* value is much better than *Translution* (-0.8). Even considering *cost*, we can not overlook such *translation quality* difference.

From the view of experienced/novice users, an experienced user may be understanding of the proper boundaries of a certain property evaluation metric, and it is useful in avoiding the deficiency of dynamic min-max boundary. While a novice user needs efforts to get such boundary and avoid this dynamical boundary problem. Those normalizations without boundary will be preferable. Thus, flexibility in the terms of normalization choices seems valuable for both experienced and novice users.

B. User Satisfaction Calculation

The goal is to evaluate the degree of matching between the QoS requirement of users and the available Web services. We note that three types of this matching, linear matchmaking [5], [8], [9], satisfaction distribution [7], and others [3]. Linear matchmaking can be viewed as a linear satisfaction distribution. Others, like the distance between QoS requirement and services [3], are not as popular. Satisfaction distribution involves how users will be satisfied with certain range of property values.

Many ways to use satisfaction distribution are studied. For example, the linear matchmaking is processed to yield a standard range (here, range $[0, 1]$), while a default satisfaction distribution over this range is used for all properties [7]. Another way is to prepare a satisfaction distribution over the preferred range of property values [22], [23]. It is also used for describing the satisfaction of preference for a group or general users [24], instead of individuals.

From the view of experienced/novice users, a novice user might accept a default satisfaction distribution (like in [7]). But for an experienced user, a controllable satisfaction distribution is more preferable. However, current described models are uncontrollable and too complex(like in [22], [23]) for experienced users to define their own satisfaction distribution.

C. Multiple Properties Aggregation

For aggregation, the key user-centered factor is the weights used for aggregating multiple QoS properties [2], [4], [6], [16], [17], [18], [19], [20]. An experienced user can be trusted with total control over all the weights. Novice users, are best served by default weight values. Because of limited knowledge, a default value is more proper for starting up. Although, there are studies mentioned the problem of weights providing. For example, multiple weights are generated from pair wise weight matrix [6], which is more or less a kind of wizard process. Still, the weight setting is assumed to be precisely provided by both the experienced users and novice users without any reference. Without experience of how the weight changes can affect the final results, it will be a challenge for the novice users.

IV. USER-CENTERED QOS COMPUTATION

Based on the above examination, we propose a user-centered QoS computation scheme by adding three components based on the general computation model (see Figure 2), so as to promote usability. We will present details below.

- Normalization Choice: when static boundaries of QoS property value is not available (like certain domain specific properties), experienced users can set metric-related expected boundaries and use the simple and efficient min-max normalization. Because the novice users can not provide such information, a new normalization will the default for them.
- Satisfaction Distribution Function: experienced users are familiar with the QoS properties, and so are able

to configure their own satisfaction distribution. Novice users are supported with default or linear satisfaction distribution (same effect as linear matchmaking).

- Weight Suggestion: experienced users can set their aggregation weight, while novice users are provided with suggested weights.

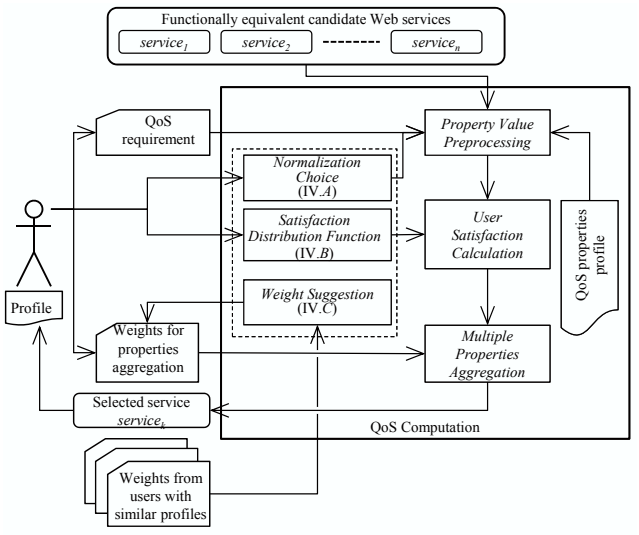


Figure 2. User-Centered QoS computation
(more details in Section IV.*A*, IV.*B*, and IV.*C*)

A. Normalization Choice

Min-max normalization is not only choice possible; other normalization techniques are available [4], [17]. Unfortunately, they do not pay special attention or assist in a situation that no static boundary are available. From this analysis, we provide the novice users with the default selection of a new normalization technique that focuses on the relative contribution of dynamic selection.

There are n Web services S=$\{s_1, s_2, \ldots, s_n\}$, the evaluated results of kth QoS property p_k from these services are $\{q_{1k}, q_{2k}, \ldots, q_{nk}\}$. Our goal is to normalize q_{ik} into $q'_{ik} \in [0,1]$ $(1 \leq i \leq n)$. If randomly selecting from them, the expectation will be the average value of them. And if one of them is selected out, the expectation will change. We use the relative change of expectation (average value)to represent the contribution of q_{ik} of s_i. Thus, the ratio of average change is calculated, when q_{ik} is selected out (see Equation (2)).

$$\begin{aligned} \textit{ratio-avg-chg}_{ik} &= \frac{\sum_j q_{jk}/n - (\sum_j q_{jk} - q_{ik})/(n-1)}{(\sum_j q_{jk} - q_{ik})/(n-1)} \\ &= \frac{1}{n}\left(\frac{q_{ik}}{(\sum_j q_{jk} - q_{ik})/(n-1)} - 1\right) \end{aligned} \tag{2}$$

However, $\textit{ratio-avg-chg}_{ik}$ is affected by n, thus we use $\textit{ratio}_{ik}$ to represent the contribution yielded by selecting q_{ik} as the target property value (see Equation (3)).

$$\textit{ratio}_{ik} = \frac{q_{ik}}{(\sum_j q_{jk} - q_{ik})/(n-1)} \tag{3}$$

Next, we normalize $\textit{ratio}_{ik} \in (0, \infty)$ into range $[0,1]$, by using the following piecewise monotonic function:

$$f(x) = \begin{cases} x/2, & \text{if } 0 \leq x \leq 1 \\ (2x-1)/2x, & \text{if } x > 1 \end{cases} \tag{4}$$

Thus, if metric is positive, the calculation is $q'_{ik} = \textit{ratio}'_{ik}$ in Equation (5), otherwise it is $q'_{ik} = 1 - \textit{ratio}'_{ik}$.

$$\textit{ratio}'_{ik} = \begin{cases} \dfrac{(n-1)q_{ik}}{2(\sum_j q_{kj} - q_{ik})}, & \text{if } q_{ik} < \sum_j q_{kj}/n \\ 1/2, & \text{if } q_{ik} = \sum_j q_{kj}/n \\ \dfrac{(2n-1)q_{ik} - \sum_j q_{kj}}{2(n-1)q_{ik}}, & \text{if } q_{ik} > \sum_j q_{kj}/n \end{cases} \tag{5}$$

We recalculate the property values (see Table III) of the example (see Table I). With regard to the *cost* normalization, it seems not as good (linear) as min-max normalization. But for *translation quality* normalization, the result is obviously more reasonable than min-max normalization. For sentence s_1, selecting *Google*(0.48) is closer to selecting *Translution* (0.52). While for sentence s_2, selecting *Translution* (0.83) is much better than selecting *Google* (0.25). Thus, we suggest this normalization for novice users.

Table III
NEW NORMALIZATION FOR EACH CANDIDATE TRANSLATION SERVICES, GOOGLE(*G*), J-SERVER(*J*), WEB-TRANSFER(*W*), AND TRANSLUTION(*T*), FOR SENTENCE s_1 AND SENTENCE s_2

Input	Property	Translation Service			
		G	*J*	*W*	*T*
s_1	cost	1	0.79	0.28	0.17
	translation quality	0.48	0.49	0.51	0.52
s_2	cost	1	0.79	0.28	0.17
	translation quality	0.25	0.39	0.63	0.83

Thus experienced users get more accurate property calculation since they provide expected boundaries for min-max normalization. While for novice users, the default choice of the above proposed normalization technique, calculates the contribution offered by dynamic selection, without demanding that the user provide boundary information of the property evaluation metric.

B. Satisfaction Distribution Function

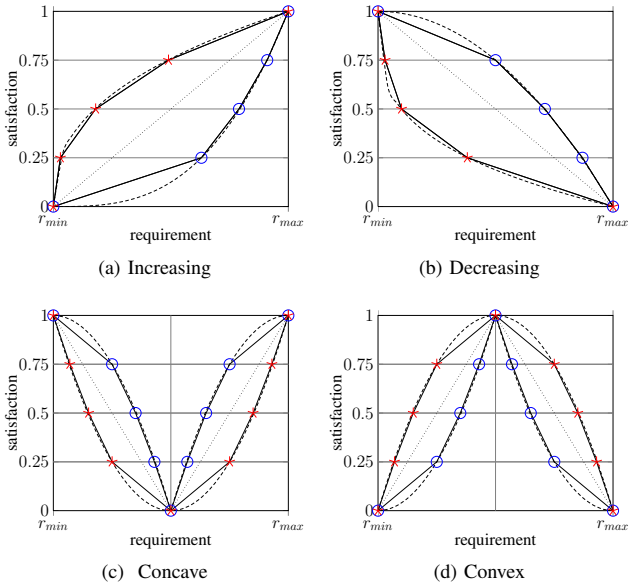
Various satisfaction distributions are possible for both general and domain specific QoS properties. For example, the satisfaction distribution of *cost* for most people is a decreasing curve [23], [24]. But a default satisfaction distribution is not enough for experience users. In dynamical task situation, more controllable mechanism is necessary. For example, considering *cost* for some people, non-free service might be acceptable in some decent tasks like in a job, but no acceptable for certain entertainment situations.

So for the same cost, two different satisfaction curve can exist. Thus, it is hard to prepare all the curves for all the people in different tasks. Then, we need a mechanism to enable the user to depict its own satisfaction distribution in a convenient manner.

We provide an approximation way to control the satisfaction distribution. First, we define the controllable reference points for users. Motivated by importance categorization works by Srivastava et al. [24] and Yau et al. [7], we categorize the satisfaction into five levels, and map them into range $[0, 1]$: *most satisfied* (1.0), *satisfied* (0.75), *medium* (0.5), *unsatisfied* (0.25), *most unsatisfied* (0).

Then, with these referent satisfaction points fixed by the users, it is easy to generate a polyline to connect these satisfaction points (see Figure 3). Based on the polyline, we can easily generate a piecewise function to approximate the potential satisfaction distribution. For example, a decreasing satisfaction distribution of $time$ for a user, could be approximated by either the star node piecewise function or circle node piecewise function, or others (see Figure 3).

For experienced users, it is required to control these five levels satisfaction points. They can flexibly approximate their own satisfaction distributions for each QoS properties, especially those domain specific ones. For novice users, if they have taken the new normalization by contribution, then a default satisfaction curve could be an angular symmetry curve with stepper changing in *medium* (0.5), similar to the default configuration proposed by Yau [7].



(a) Increasing (b) Decreasing

(c) Concave (d) Convex

Figure 3. A polyline way to approximate satisfaction distribution (each *Increasing* (a) and *Decreasing* (b) type distribution is approximated by five controllable reference points, others are split into this two types, such as *Concave* (c) and *Convex* (d))

C. Weight Suggestion

Though most existent computation assumes that users have the ability to provide the aggregation weight. This will support experienced users but is too complex for novice users, who are unable to predict the effect of their choices.

We propose the weight suggestion scheme for novice users that refers the settings made bys experienced users(see Figure 4). User profile similarity has been extensively researched and adopted in Web service area [25], [26], [27]. Here we assume the user profile similarity function is available.

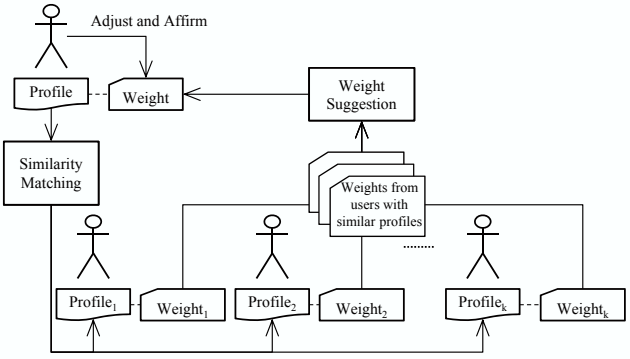


Figure 4. A way of weight suggestion

We adopt the distribution strategy to calculate the weight suggestion. As Srivastava et al., who hypothesized that a group of people have certain rate distribution [24], we hypothesize that users with similar profiles will have similar weight setting distribution for each QoS property. Because the number of people with similar profile can be limited, we use a t-student distribution.

For the k-th QoS property p_k, its corresponding weight is w_k. When the weight settings on this property p_i of n users are available, they are $\{w_{k1}, w_{k2}, \ldots, w_{kn}\}$. We need to calculate the expectation μ_k as the suggested default value for the novice user.

According to t-student, we can calculate t_k, s_k (see Equation 6, 7). We then calculate the confidence interval of the expectation μ_k (see Equation 8). Given probability $1 - \alpha$, we can provide the reference range to w_k for novice users.

$$t_k = \frac{\bar{w_k} - \mu_k}{s}\sqrt{n-1}, \text{ here } \bar{w_k} = \frac{\sum_1^n w_{ki}}{n} \tag{6}$$

$$s_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(w_{ki} - \bar{w_k})^2} \tag{7}$$

$$P(\bar{w_k} - t_{\frac{\alpha}{2}}\frac{s_k}{\sqrt{n-1}} < \mu_k < \bar{w_k} + t_{\frac{\alpha}{2}}\frac{s_k}{\sqrt{n-1}}) = 1 - \alpha \tag{8}$$

This scheme provides each novice user with a weight suggestion. They are then able to adjust the weight setting

in a very easy manner if they so desire. If the novice user adjusts the weight, we can easily normalize the weight setting into range $[0, 1]$ by $w'_k = w_k / \sum w_i$.

V. Case Study

We study the case of QoS calculation in selecting a translation service. Many machine translation services are available from the service-oriented platform Language Grid [28]. Faced with these functionally equivalent Web services, users have to select their preferred one based on the non-functional properties, such as *time* and *cost*. Moreover, the domain specific property *translation quality* is a key that users pay much attention to. Then, this QoS computation includes not only the general properties of *time*, *cost*, but also the domain specific property of *translation quality*. We will explain how the proposed mechanism works for translation service selection.

A. Configuration

There are 5 groups users, including *Document* translation users, *Chat* translation users, *Web* translation users, and 2 groups of test users. Each group has 4 people. And we describe the different expectations of the properties of translation service as follows.

Document Translation Expectation:

- *translation quality* is most important for formal document translation. Here, evaluation is by *BLEU* score, which suits documents. It should be higher than certain threshold. Otherwise, it becomes unacceptable exponentially.
- *time* and *cost* have a linear impact on service quality.

Chat Translation Expectation:

- *time* is most important for instant chat translation. It should shorter than a certain threshold. Otherwise, it becomes unacceptable exponentially.
- *translation quality* is evaluated in *WER* score, which is most popular for spoken language translation.
- *translation quality* and *cost* have a linear impact on service quality.

Web Translation Expectation:

- *cost* is most important for free Web browsing. Free is preferred. Any cost raise the level of unacceptability exponentially.
- *translation quality* is evaluated in *NIST* score, which emphasize correctly translating key words.
- *translation quality* and *time* have a linear impact on service quality.

We train *Document*, *Chat* and *Web* translation users into experienced users, and treat the 2 groups of reserved users as novice users. The train process includes two practices. First, each group member has to manually select a best service from four translation services. Second, each member has to configure the user-centered QoS calculation for selection and manually check the results, so as to master configuration.

After traning, we have 3 group experienced users. 50 translation sources are prepared for each group. The *translation quality*, *time* were collected from four translation services, with the preset *cost*. Then we collect the QoS properties of each translation services for each translation source. The *manually selection* of best translation result by each experienced user is used as the standard selection result. Here, we assume that a user-centered QoS computation should replicate the service selection made by experts.

B. Evaluation

After we got all the QoS property values for all translation sources from the three groups, and the standard selection result by human from each experienced user, we evaluated the *HitRate* of each configured QoS computation. *HitRate* is the recall of translation result selected by the QoS computation compared to the standard selection result. For example, for one experienced user configured QoS computation, if 50 are automatically selected, and 30 of them are found in the standard selection result of the same translation sources, then its *HitRate* is 30/50 = 60%.

We analyze two comparisons between the QoS computation customized by experienced users and novice uers.

- Comparing the HitRate of a QoS computation configured by an inner group experienced user to an outer group experienced user. This indicates whether the participation of users is important or not.
- Comparing the HitRate of a QoS computation randomly configured by a novice user to the default QoS computation by our proposed way. This indicates whether the suggestion is useful or not.

Table IV
Average HitRate of user configured service selection by experienced users of 3 groups (*Document*, *Chat*, and *Web*), and novice users of 2 groups (*Random* configuration, and *Suggested* configuration)

Average HitRate of User Configured Service Selection	Manually Selection		
	Document	*Chat*	*Web*
Document Experienced Users	87%	56%	62%
Chat Experienced Users	51%	76%	65%
Web Experienced Users	60%	61%	81%
Random Novice Users	56%	51%	64%
Suggested Novice Users	73%	68%	69%

C. Analysis

For each experienced user, his/her configured QoS computation yielded selection results (translation service) over three different groups of translation sources. The selection results are compared to the standard selection result of the same translation source, and HitRate of each pair is calculated. We can then calculate the Average HitRate (see Table IV). Obviously, the inner group experienced user

received higher HitRate over the standard data in the same group. For example, consider the experience users in *Document* group; their configured QoS computation achieved the highest score (87%) for the translation sources of *Document*, compared to *Document* standard results. It shows that one configured QoS computation cannot meet all expectations. Thus, user participation is important, and the selection will be more efficient once experienced users have generated proper setting for user-centered QoS computation.

For each novice user in one group, his/her randomly configured QoS computation yielded service selection for three difference groups of translation sources. The *Average HitRate* of this group is lower than the other group novice users who took advantage of suggested configuration (see Table IV). For example, for the same standard data of *Document*, a random configuration received lower *Average HitRate* (56%), 31% lower than the experience users (87%). However, the suggested configuration yielded *Average HitRate* of 73%, not as higher as the achieved by experienced users (87%). Thus, the suggested configurations are very effective for novice users, and are far superior to random configuration.

VI. Related Works

In the field of QoS based Web service selection, various subjects have been intensely researched: For examples, Menasc et al. [22] extended QoS broker with a consideration of utility functions. QoS broker is an important topic describing the architecture of QoS based web service selection. Yu et al. [29] provided a constraint view of QoS composition. QoS preference and constraint are major aspects of QoS composition. Zhou et al. [30] provided QoS ontology language for QoS based Web service matchmaking. In this paper, we focus on the QoS computation, and so do not address QoS composition description, QoS broker, QoS matchmaking, etc. In the QoS based Web service selection, when the QoS computation becomes necessary, it will be easy to extend our proposed mechanism to cover those aspects.

In the field of QoS computation, many studies have gone beyond the examined general QoS computation steps: Liu, et.al. [4] described bounded dynamically selection computation, which still requires experienced-based maximum normalized value setting. Different from dynamical selection, Hang and Singh [23] adopted quality distribution to select a Web service with best quality expectation. Wang et al. [5] showed interest in the semantic description of QoS. Besides semantic description, Vu et al. [2] paid attention to reputation-based comparison of QoS. Yau et al. [7] proposed a requirement specification to standardize expression of user requirements. In this paper, we focus on the user-centered perspective of QoS computation. Different from the above works and their attention to functional enhancement, we pay attention to the usability of QoS computation. Thus, we can integrate our proposal into a wide range of functional enhancements.

In user-centered Web service area, Balke and Wagner [10] suggested an individual adaptation on service provisioning. Mobedpour and Ding [3] described a wizard for requirement query formulation. Liu et al. [12] adopted personalized perspective to present service discovery requirement. In this paper, our concept is to differentiate experienced from novice users, and so enhance usability for both.

VII. Conclusion

We studied general QoS computation for Web service selection with its three main steps: QoS property calculation, satisfaction calculation, and multiple properties aggregation. From the user-centered perspective, developed from an understanding of experienced/novice users, we identified several usability defects in each step. We introduced user-centered QoS computation by promoting the participation of users; our techniques provide choices and suggestions to promote the facility and flexibility for both experienced and novice users.

For normalization choices in property value preprocessing, we defined a new normalization method for novice users, who lack the experience needed to provide expectation boundaries. Experienced users, on the other hand, are provided with the freedom to set expectation boundaries, and receive easier-to-apply min-max normalization results.

As the satisfaction distribution function in the satisfaction calculation step, we provided a polyline approximation of common satisfaction distributions for experienced users. It allows easy control of the entry of personal satisfaction preferences. For novice users, a default configuration is suggested.

For weight suggestion, we employ the t-student distribution strategy, which calculates range references by taking advantage of settings entered by experienced users. Experienced users, with their rich experience, can optimize already useful weight settings.

Finally, we did a case study in translation service selection. Experience users could achieve higher HitRate through their optimized configurations, while novice users achieved better HitRate by accepting the suggested configuration. It shows that, our user-centered QoS computation is both necessary and useful.

Acknowledgment

This research was partially supported by Service Science, Solutions and Foundation Integrated Research Program from JST RISTEX.

References

[1] K. Kritikos and D. Plexousakis, "Requirements for qos-based web service description and discovery," *IEEE Trans. Serv. Comput.*, vol. 2, pp. 320–337, October 2009.

[2] L.-H. Vu, F. Porto, K. Aberer, and M. Hauswirth, "An extensible and personalized approach to qos-enabled service discovery," in *Proceedings of the 11th IDEAS*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 37–45.

[3] D. Mobedpour and C. Ding, "User-centered design of a qos-based web service selection system," *Service Oriented Computing and Applications*, pp. 1–11, 2011.

[4] Y. Liu, A. H. Ngu, and L. Z. Zeng, "Qos computation and policing in dynamic web service selection," in *Proceedings of the 13th international WWW Alt*. New York, NY, USA: ACM, 2004, pp. 66–73.

[5] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, "A qos-aware selection model for semantic web services," in *Proceedings of the 4th ICSOC*. Springer, 2006, pp. 390–401.

[6] J. Cao, J. Huang, G. Wang, and J. Gu, "Qos and preference based web service evaluation approach," in *Eighth International Conference on Grid and Cooperative Computing, (GCC 2009).*, 2009, pp. 420–426.

[7] S. S. Yau and Y. Yin, "Qos-based service ranking and selection for service-based systems," in *Proceedings of the 2011 IEEE International Conference on Services Computing (SCC 2011)*, Washington, DC, USA, 2011, pp. 56–63.

[8] M. Comuzzi and B. Pernici, "A framework for qos-based web service contracting," *ACM Trans. Web*, vol. 3, pp. 10:1–10:52, July 2009.

[9] Q. Ma, H. Wang, Y. Li, G. Xie, and F. Liu, "A semantic qos-aware discovery framework for web services," in *IEEE International Conference on Web Services (ICWS 2008)*, sept. 2008, pp. 129–136.

[10] B. W. and W. M., "Cooperative discovery for user-centered web service provisioning," in *In IEEE International Conference on Web Services (ICWS 2003)*, 2003, pp. 191–197.

[11] W. Rong, K. Liu, and L. Liang, "Towards personalized ranking in web service selection," in *IEEE International Conference on e-Business Engineering, (ICEBE 2008).*, oct. 2008, pp. 165–172.

[12] X. Liu, G. Huang, and H. Mei, "Discovering homogeneous web service community in the user-centric web environment," *Services Computing, IEEE Transactions on*, vol. 2, no. 2, pp. 167–181, april-june 2009.

[13] D. Mobedpour, C. Ding, and C.-H. Chi, "A qos query language for user-centric web service selection," in *IEEE International Conference on Services Computing (SCC 2010)*, july 2010, pp. 273–280.

[14] A. Bramantoro and T. Ishida, "User-centered qos in combining web services for interactive domain," in *Fifth International Conference on Semantics, Knowledge and Grid (SKG 2009)*, oct. 2009, pp. 41–48.

[15] S. Goto, Y. Murakami, and T. Ishida, "Reputation-based selection of language services," in *IEEE International Conference on Services Computing (SCC 2011)*, july 2011, pp. 330–337.

[16] G. Canfora, M. D. Penta, R. Esposito, F. Perfetto, and M. L. Villani, "Service composition (re)binding driven by application-specific qos," in *Proceedings of the 4th ICSOC*, 2006, pp. 141–152.

[17] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction for web services via collaborative filtering," in *IEEE International Conference on Web Services (ICWS 2007)*, july 2007, pp. 439–446.

[18] E. Al-Masri and Q. Mahmoud, "Qos-based discovery and ranking of web services," in *Proceedings of 16th International Conference on Computer Communications and Networks (ICCCN 2007)*, aug. 2007, pp. 529–534.

[19] Q. Li-li and C. Yan, "Qos ontology based efficient web services selection," in *International Conference on Management Science and Engineering (ICMSE 2009).*, sept. 2009, pp. 45–50.

[20] A. F. M. Huang, C.-W. Lan, and S. J. H. Yang, "An optimal qos-based web service selection scheme," *Inf. Sci.*, vol. 179, pp. 3309–3322, September 2009.

[21] H. Muñoz Frutos, I. Kotsiopoulos, L. M. Vaquero Gonzalez, and L. Rodero Merino, "Enhancing service selection by semantic qos," in *Proceedings of the 6th ESWC*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 565–577.

[22] D. Menasce and V. Dubey, "Utility-based qos brokering in service oriented architectures," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*, july 2007, pp. 422–430.

[23] C.-W. Hang and M. P. Singh, "From quality to utility: Adaptive service selection framework," in *ICSOC*, 2010, pp. 456–470.

[24] A. Srivastava and P. Sorenson, "Service selection based on customer rating of quality of service attributes," in *IEEE International Conference on Web Services (ICWS 2010)*, july 2010, pp. 1–8.

[25] B. Smyth and P. Cotter, "A personalized television listings service," *Commun. ACM*, vol. 43, pp. 107–111, August 2000.

[26] K. Sugiyama, K. Hatano, and M. Yoshikawa, "Adaptive web search based on user profile constructed without any effort from users," in *Proceedings of the 13th international conference on World Wide Web*, ser. WWW '04. ACM, 2004, pp. 675–684.

[27] J. Golbeck, "Trust and nuanced profile similarity in online social networks," *ACM Trans. Web*, vol. 3, pp. 12:1–12:33, September 2009.

[28] T. Ishida(eds.), *The Language Grid: Service-Oriented Collective Intelligence for Language Resource Interoperability*. Springer, 2011.

[29] T. Yu, Y. Zhang, and K. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM TWEB*, vol. 1, 2007.

[30] C. Zhou, L.-T. Chia, and B.-S. Lee, "Daml-qos ontology for web services," in *Web Services, 2004. Proceedings. IEEE International Conference on*, july 2004, pp. 472–479.