An agent-based web service selection and ranking framework with QoS

Guobing Zou[1], Yang Xiang[1], Yanglan Gan[1], Dong Wang[1] and Zengbao Liu[2]
[1] *Department of Computer Science and Technology, Tongji University, Shanghai, 201804, China*
[2] *Dongtan Coal Mine, Yanzhou Coal Mine Co.,Ltd, Zoucheng, 273500, China*
Email: guobing278@sina.com

Abstract

With the number of services published over the Internet growing at an explosive speed, it is difficult for service requesters to select satisfactory web services, which provide similar functionalities. Quality of service (QoS) is considered the most important non-functional criterion for further service filtering. In this paper, we firstly give a web service description model that considers service QoS information, and then present an overall service selection and ranking framework with QoS (WSSR-Q) based on previous service description model. Finally, service selection algorithm, ranking algorithm, and quality updating mechanism are proposed in detail concerning QoS attributes value, respectively. Simulation experiments demonstrate that proposed framework with QoS for service selection and ranking can satisfy service requesters' non-functional information requirements and achieve better web service selection effectiveness.

1. Introduction

Web services are self-describing software entities that can be advertised, located and used over the Internet using a set of standards such as UDDI, WSDL, and SOAP. Web service technology is becoming more and more popular in many practical application domains [1], such as electronic commerce, flow management, application integration, etc. It presents a promising solution for solving platform interoperability problems encountered by the application system integrators. With the rapid development of web service technology in these years, traditional XML based standards (i.e., UDDI) have been mature during service registry and discovery process.

In the traditional discovery model for web services, UDDI registry allows service providers to register their web services via tModel, which is a form of metadata providing a reference system for service information [2]. Furthermore, web service requesters submit their service requirement to UDDI registry that is in charge of matching requirement with advertised services. However, with the mushrooming of various web services on the Internet, multiple web services will be discovered by traditional matching engine with similar functionalities. Consequently, it becomes difficult for service requesters to find the most appropriate web service by their own subjective judgments. In this scenario, quality of service (QoS) can be considered as a second criterion for web service selection.

In this paper, we develop an efficient approach to implement optimal web service selection and ranking for fulfilling service requesters' functional and non-functional requirements. Our work is distinguished from other related research in the following three aspects. Firstly, we give a web service description model for describing web services where non-functional attributes are taken into account. Secondly, an overall framework of service selection and ranking with QoS is proposed based on previous given description model. Thirdly, we respectively concentrate on a specific service selection algorithm, a service ranking algorithm, and quality updating mechanism.

The remainder of the paper is organized as follows. Section 2 reviews related works about service discovery and selection. Section 3 presents a web service description model. The general framework for web service selection and ranking is proposed in Section 4. Service selection algorithm, service ranking algorithm and quality updating mechanism are proposed respectively in Section 5. Related simulation experiment has been conducted to validate and evaluate the effectiveness of proposed approach in Section 6. Finally, section 7 concludes the paper.

2. Related work

The work was proposed in the reference [3], which presented a model of reputation-enhanced QoS-based web service discovery that combines an augmented UDDI registry to publish the QoS information and a reputation manager to assign reputation scores to services. However, it only described an abstract service matchmaking, ranking and selection algorithm. Moreover, they failed to give an efficient metrics method for QoS computation, which was only evaluated by the dominant QoS attribute. In order to enable quality-driven web service selection, the authors in [4] proposed a QoS computation model by implementation and experimentation with a QoS registry in a hypothetical phone service provisioning. Unfortunately, as a result of

978-1-4244-4520-2/09/$25.00 ©2009 IEEE

their measurement way of QoS values normalization, it is very difficult to make a uniform evaluation for all quality criteria because their QoS metrics values are not limited in a definite range. Therefore, it will bring about a problem that a quality attribute even has a higher weight, while its internal impact is decreased by its smaller QoS value.

In [5], the authors presented a QoS-based service selection model. They specified QoS ontology and its vocabulary by Web Service Modeling Ontology (WSMO) [6]. Especially, they gave a selection mechanism based on an optimum normalization algorithm, which integrates service selection and ranking. Although this can simplify computational complexity, it will also cause a problem that some returned web services with high synthetic QoS score can not fulfill some single QoS criteria condition. In [7], the authors proposed a web service discovery model where functional and non-functional requirements are taken into account. However, not any feedback can be collected from service requesters as reference to updating QoS value.

To address these problems, this paper proposes an agent-based web service selection and ranking framework concerning QoS information. Our goal of this work is to help service requesters select the most satisfactory web services fulfilling their QoS requirements.

3. Web service description model

In order to facilitate providers to publish service information with QoS, it is necessary to model service description, as well as provides a mechanism for requesters to submit service requirements. An efficient web service description model has been given in [8]. However, it does not include QoS registry. We propose a service description model concerning QoS called *WSDM-Q*, which contains two parts of definitions: web service and service request.

Definition 3.1 **Web service.** A web service in web service repository is defined as a five tuple:

$$ws=\{ServiceKey, wsName, wsDesp, Q_P, OprSet\} \quad (1)$$

- *ServiceKey* is the unique identifier;
- *wsName* represents web service name;
- *wsDesp* is service functional description;
- Q_P is published QoS information that is denoted as $Q_P=Q_N \cup Q_D$. Where Q_N represents necessary quality criteria set for all web services and Q_D represents domain-specific quality criteria set for specific web services.
- *OprSet* is web operation set denoted as $OprSet=\{opr_1, opr_2, \ldots, opr_s\}$. Where each $opr_i(1 \le i \le s)$ can be executed for a specific function task.

Similarly, for the requirements of service requester, we give a corresponding service request description.

Definition 3.2 **Service request.** A service request is defined as a four tuple:

$$sq=\{wsName, InSet, OutSet, Q_R\} \quad (2)$$

Where, *wsName*, *InSet* and *OutSet* have the same meaning as in *Definition 3.1* and *3.2*. The difference is that these are the request information. Q_R includes necessary and domain-specific quality criteria set, which is defined as $Q_R=Q_N \cup Q_D$ similar with the *Definition 3.1*.

WSDM-Q is used to publish web services or submit service requirements in the following framework.

4. Service selection and ranking framework

There are three roles in traditional service discovery model, these are service provider, service consumer and UDDI registry as it appears in Figure 1.

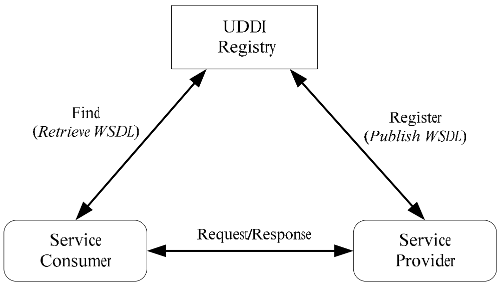

Figure 1. Traditional web service publish-find-bind model

Although UDDI registries have been widely adopted, when a set of services satisfying consumer's functional requirements have been discovered, it is hard for service consumers to make decision which one will be eventually invoked among these services with similar capabilities. Because it lacks of further service filtering and selection.

In our framework, we extend the overall architecture proposed to support service selection, ranking and quality updating, which consists of web services repository, QoS database, service selection module, quality rating database, service provider and service requester. The general framework for web service selection and ranking with QoS called *WSSR-Q* is shown in Figure 2.

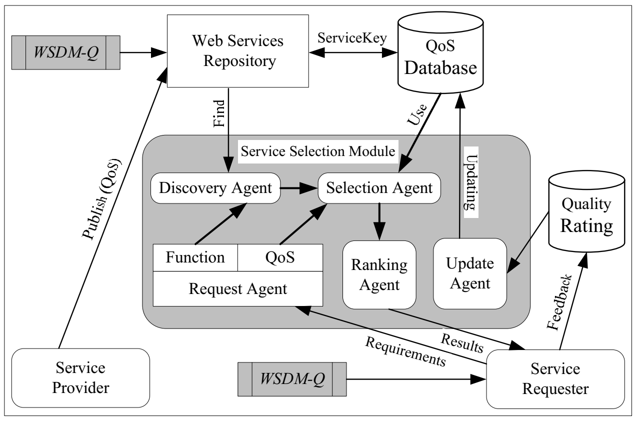

Figure 2. The general framework of web service selection and ranking with QoS

The kernel of our framework lies in the Service selection module, which is a steering infrastructure and involves five correlative agents as follows:

- The Request Agent which provides interface and communicates with service requester for acquiring functional requirements and QoS constraints.
- The Discovery Agent which is in charge of finding initial web service set satisfying service requester's functional requirements.
- The Selection Agent which collects QoS information from QoS database in terms of initial discovered web service set and then selects web service set fulfilling service requester's QoS constraints.
- The Rank Agent which is utilized to calculate synthetic QoS score of each selected web services, and then ranks them in a descending sequence according to their QoS marks. Finally, ranked service set is returned back to service requester.
- The Update Agent which refreshes quality criteria value in the QoS database according to accumulated feedback information in quality rating database.

For other components in our framework, web services repository provides registry mechanism for web service providers who publish service functional information, as well as supply non-functional information. Especially, published QoS information is stored in QoS database correlative with web services repository by the ServiceKey. Quality rating database accumulates all the QoS feedback information from service requesters' invoking services.

5. Web service selection, ranking and updating

Some classic web service discovery algorithms have been proposed, such as in reference [2, 8], which is out of the scope in this paper. Therefore, the task of discovering web services in term of requesters' functional requirements is performed by the existed service matchmaking engine.

5.1 Service selection algorithm

In order to illustrate our method, we give some notations used in the following subsections of this paper as it appears in Table 1:

Table 1. Notations definition and description

Notations	*Explanations*
S	Web services repository
s_i	A web service, $s_i \in S$
S_D	Discovered service set, $S_D \subseteq S$
S_S	Selected service set, $S_S \subseteq S_D$
S_R	Ranked service set, $S_R \subseteq S_S$
q_{ij}	QoS name at position j of s_i
v_{ij}	Constraint value of q_{ij}
Q_P^i	Published QoS information set of s_i
Q_R	Submitted QoS requirement set
c_{ij}	A constraint relation at position j of s_i
c_k	Request ternary relation at position k

Definition 5.1 **Ternary constraint relation.** A QoS ternary constraint relation is defined as $c(q, op, v)$.

Where, q represents quality attribute name, v gives constraint value, and op is constraint operator between q and v. constraint operator set $\{\le, =, \ge\}$ is used in this paper.

For service providers, they publish web services' QoS information. For each service s_i, its QoS information set is composed of several QoS ternary constraint relations.

$$Q_P^i=\{(c_{i1}(q_{i1},op_{i1},v_{i1}),\ldots,c_{im}(q_{im},op_{im},v_{im})),(c_{i(m+1)}(q_{i(m+1)},op_{i(m+1)},v_{i(m+1)}),\ \ldots,\ c_{ih}(q_{ih},op_{ih},v_{ih}))\} \tag{3}$$

Q_P^i consists of Q_N and Q_D, where each dimension is a QoS ternary constraint relation. Q_N and Q_D contain m and (h-m) QoS ternary constraint relations respectively. At the same time, the constraint operator ' = ' is utilized in $c_{ij}(i \in N, 1 \le j \le h)$ to publish QoS information.

On the other side, service requesters submit their QoS requirement set, which is formalized as:

$$Q_R=\{(c_1(q_1,op_1,v_1),\ldots,c_m(q_m,op_m,v_m)),(c_{m+1}(q_{m+1},op_{m+1},v_{m+1}),\ \ldots,\ c_n(q_n,op_n,v_n))\} \tag{4}$$

Similarly, Q_R consists of Q_N and Q_D. Especially, constraint operator set $\{\le,\ge\}$ is adopted in $c_k(1 \le k \le n)$ for service requesters to submit their QoS requirements.

We assume that t services with similar functionality are discovered from web services repository S by the Discovery Agent, denoted as $S_D=\{s_1, s_2, \ldots, s_t\}$.

The description of Service selection algorithm with QoS (*SSA-Q*) is shown in Algorithm 1.

```
Algorithm 1: Service selection with QoS (SSA-Q).
Input: $S_D$ and QoS requirement $Q_R$;
Output: Selected web service set $S_S$;
1.  $S_S \leftarrow S_D$;
2.  If $Q_R$=NULL then
3.      Return $S_S$;
4.  Else if $Q_N \ne$ NULL then { //$Q_N \subseteq Q_R$
5.      $len \leftarrow Q_N.length$;
6.      $S_S \leftarrow SelectWithQoS(S_D, Q_N, len, 1)$; }
7.  If $S_S \ne$ NULL then {
8.      $len \leftarrow Q_D.length$; //$Q_D \subseteq Q_R$
9.      $S_S \leftarrow SelectWithQoS(S_S, Q_D, len, 0)$; }
10.     Return $S_S$; }
11. Else
12.     Return NULL
```

```
SelectWithQoS(S_X, Q_X, lenX, kindX).
1.  $S_{res} \leftarrow$ NULL; //returned service set
2.  For $u \leftarrow 1$ to $S_X.length$ do {
3.      $Q_P^u \leftarrow S_X[u].Q_P$;
4.      $counter \leftarrow 0$;
5.      For $w \leftarrow 1$ to $Q_X.length$ do {
6.          $c_w(q_w,op_w,v_w) \leftarrow Q_X[w]$;
7.          $c_{ij}(q_{ij},op_{ij},v_{ij}) \leftarrow findTerRel(Q_P^u,q_w)$;
8.          If $(c_{ij}(q_{ij},op_{ij},v_{ij})$=NULL $\wedge kindX=1)$ then break;
9.          Else If $(c_{ij}(q_{ij}, op_{ij}, v_{ij})$=NULL$\wedge kindX$=0$)$ then
10.             $counter \leftarrow counter + 1$;
11.         Else If $(c_{ij}(q_{ij}, op_{ij}, v_{ij}) \neq$ NULL$)$ then {
12.             If $(op_w.equals('\leq') \wedge v_w \geq v_{ij})$ then
13.                 $counter \leftarrow counter + 1$;
14.             Else If $(op_w.equals('\geq') \wedge v_w \leq v_{ij})$ then
15.                 $counter \leftarrow counter + 1$;
16.             Else break; }
17.         If $(counter$=$lenX)$ then //$S_X[u]$ judgment
18.             $S_{res}.append(S_X[u])$; }
19. }
20. Return $S_{res}$;
```

In algorithm 1, if Q_R is not specified (*line* 2), original S_D is returned as selected service set S_S. otherwise, when Q_N is specified (*line* 4), *SelectWithQoS* (*line* 6) is executed and returns an initial selection result to S_S. If there are returned services in S_S satisfying Q_N requirements (*line* 7), *SelectWithQoS* (*line* 9) is executed again to select services and store in S_S according to Q_D and initial result set.

In the function of ***SelectWithQoS***, we loop candidate service set S_X (*line* 2) and compare its QoS information with Q_X. For each service $S_X[u]$, its QoS information is stored in Q_P^u (*line* 3). For each ternary constraint relation $c_w(q_w,op_w,v_w)$ (*line* 6) in Q_X, it is respectively handled by a lookup function $findTerRel(Q_P^u, q_w)$ (*line* 7) that returns a corresponding $c_{ij}(q_{ij},op_{ij},v_{ij})$ with same quality attribute name or NULL. When there is no matched c_{ij} (*line* 8-10), the current service can not meet Q_X in case of *kindX*=1 (Q_X=Q_N), otherwise, counter is increased by one if *kindX*=0 (Q_X=Q_D). When there is a matched c_{ij} (*line* 11-16), counter is increased if $(op_w \text{ is } '\leq' \wedge v_w \geq v_{ij})$ or $(op_w \text{ is } '\geq' \wedge v_w \leq v_{ij})$. Finally, service's QoS satisfiability is judged by comparing *counter* and *lenX* (*line* 17-18).

5.2 Service ranking algorithm

After the service selection process, *r* web services are picked out from S_D by the Selection Agent. The selected service set is denoted as S_S={s_1, s_2, …, s_r}. Service ranking algorithm with QoS (*SRA-Q*) is shown in Alogrithm 2.

Algorithm 2: *Service ranking with QoS (SRA-Q).*
Input: S_S, Q_R and quality criteria weight array *W*;
Output: Ranked web service set S_R;
Step 1: generate quality criteria matrix M_S.

```
1.  $M_S \leftarrow$ NULL;
2.  For $i \leftarrow$ 1 to $r$ do {
3.      $Q_P^i \leftarrow S_S[i].Q_P$;
4.      For $j \leftarrow$ 1 to $Q_R.length$ do {
5.          $c_j(q_j, op_j, v_j) \leftarrow Q_R[j]$;
6.          $c_{uw}(q_{uw}, op_{uw}, v_{uw}) \leftarrow findTerRel(Q_P^i, q_j)$;
7.          If $c_{uw}(q_{uw}, op_{uw}, v_{uw}) \neq$ NULL then
8.              $M_S[i,j] \leftarrow v_{uw}$;
9.          Else $M_S[i,j] \leftarrow 0$; }
10. }
```

Step 2: generate normalized quality criteria matrix $M_S^{'}$.

```
1.  $M_S^{'} \leftarrow$ NULL;
2.  For $j \leftarrow$ 1 to $Q_R.length$ do {
3.      $q_{\max} \leftarrow Max_{k=1}^{r}\{M_S[k,j]\}$;
4.      $q_{\min} \leftarrow Min_{k=1}^{r}\{M_S[k,j]\}$;
5.      For $i \leftarrow$ 1 to $r$ do {
6.          If $Q_R[j].op_j.equals('\geq')$ then
7.              $M_S^{'}[i,j] \leftarrow \sqrt{M_S[i,j]\text{-}q_{min}} \Big/ \sqrt{q_{max}\text{-}q_{min}}$;
8.          Else If $Q_R[j].op_j.equals('\leq')$
9.              $M_S^{'}[i,j] \leftarrow \sqrt{q_{max}\text{-}M_S[i,j]} \Big/ \sqrt{q_{max}\text{-}q_{min}}$; }
10. }
```

Step 3: calculate and rank each service's QoS value.

```
1.  For $i \leftarrow$ 1 to $r$ do {
2.      $qSum_i \leftarrow \sum_{k=1}^{n}(w_k * M_S^{'}[i,k])$;
3.      $S_R.rank(qSum_i, s_i)$; }
4.  Return $S_R$;
```

In the first step, Q_R is taken as benchmark for yielding n columns and r rows are formed by each candidate service s_i (1≤*i*≤*r*). Each row represents a candidate service, and each column contains QoS values of a quality attribute in a Q_R's ternary constraint relation. i.e., qos_{ij} is generated by service s_i and $c_j(q_j,op_j,v_j)$. If there exists a $c_{uw}(q_{uw},op_{uw},v_{uw})$ sharing the same quality name with c_j, v_{uw} is used as qos_{ij}'s value. Otherwise, qos_{ij} is set 0. The generated quality criteria matrix M_S is shown in the following equation.

$$M_S = \begin{pmatrix} qos_{11} & qos_{12} & \cdots & qos_{1n} \\ qos_{21} & qos_{22} & \cdots & qos_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ qos_{r1} & qos_{r2} & \cdots & qos_{rn} \end{pmatrix} \qquad (5)$$

In the second step, each matrix element in M_S is normalized with metrics function and mapped in the range of [0,1]. For each qos_{ij}, its normalized value qos'_{ij} is calculated by q_{max} and q_{min} (maximum/minimum value in column *j*). More specifically, when constraint operator of c_j in Q_R equals '≥' (*line* 6-7), all QoS values of column j are normalized in a monotonically increasing way. Otherwise, it is measured in a monotonically decreasing way (*line* 8-9). The generated matrix $M_S^{'}$ is shown as follows.

$$M_S^{'} = \begin{pmatrix} qos_{11}^{'} & qos_{12}^{'} & \cdots & qos_{1n}^{'} \\ qos_{21}^{'} & qos_{22}^{'} & \cdots & qos_{2n}^{'} \\ \vdots & \vdots & \vdots & \vdots \\ qos_{r1}^{'} & qos_{r2}^{'} & \cdots & qos_{rn}^{'} \end{pmatrix} \qquad (6)$$

In the third step, for each candidate service s_i (1≤*i*≤*r*) in S_S, its synthetic QoS value is calculated based on weight array *W* and normalized quality criteria matrix where each row corresponds to a service. Then, we rank and append them into S_R according to their comprehensive QoS marks. Finally, S_R is returned to service requester.

5.3 Quality updating mechanism

The Update Agent collects all requesters' feedback information, calculates comprehensive value, and revises new QoS value to QoS database. More specifically, for a service s_i ($1 \leq i \leq r$) invoked by requester, there are h QoS ternary constraint relations $c_{ij}(q_{ij},op_{ij},v_{ij})$ in the equation (4), where $1 \leq j \leq h$. We assume that L requesters have invoked s_i during a specific time period and have given their QoS feedback values for quality attribute q_{ij}: qfd_1, qfd_2, …, qfd_L. The updating value of q_{ij} is defined as:

$$v_{ij}' \leftarrow w_s \times v_{ij} + (1\text{-}w_s) \times \sum_{k=1}^{L} \frac{qfd_k}{L} \tag{7}$$

In the above formula, the updating value is calculated by original QoS value v_{ij} and mean value of L requesters' feedback information. Where, weight factor w_s reflects the importance of original QoS value.

6. Simulation experiment

In order to validate the effectiveness of our proposed framework and explain its execution process, we have simulated a set of services' QoS information in Table 2, where most parts of its quality criteria are based on test data in [4]. Especially, each column's corresponding value types are {dollar, real of (0, 1], microsecond, microsecond, real of (0, 5], real of [0, 1], real of [0, 1]}.

Table 2. A set of QoS information of service providers

wsName	Price	Avail	TimeOut	Execu	Repu	ComRat	PenRate
ABG	25	0.7	75	100	3.0	0.5	0.5
BTC	40	0.85	200	40	2.5	0.8	0.1
WS_1	46	Null	65	60	1.0	0.7	0.4
WS_2	38	0.8	120	25	3.5	0.85	0.3
WS_3	27	0.9	95	30	4.0	Null	0.1
WS_4	30	0.75	180	85	3.0	0.95	0.2

In Table 2, each candidate service in S_D={*ABG*, *BTC*, WS_1, WS_2, WS_3, WS_4} satisfies functional requirement.

During the process of service selection, we simulate a service requester whose QoS requirement is denoted as:

Q_R={(c_1(*Price*, $\leq$,50),c_2(*Availability*, $\geq$,0.75),c_3(*TimeOut*, $\geq$, 70), c_4(*Compensation rate*, $\geq$,0.7), c_5(*Penalty rate*, $\leq$,0.45)), (c_6(*Execution duration*, $\leq$,100),c_7(*Reputation*, $\geq$,2.5))}.

In the QoS requirement Q_R, Q_N={c_1, c_2, c_3, c_6, c_7} and Q_D={c_4, c_5}. After the execution of selection algorithm *SSA-Q*, we filter selected service set S_S={*BTC*, WS_2, WS_3, WS_4}. Especially, *ABG* and WS_1 are not selected because they fail to pass the requester's QoS requirement condition of {c_2, c_4, c_5} and {c_2, c_3, c_7}, respectively.

Service ranking algorithm *SRA-Q* is executed after *SSA-Q* to generate ranked service set containing three steps. Firstly, generated quality criteria matrix M_S is shown in the equation (8) according to the step one of the algorithm

$$M_S = \begin{pmatrix} 40 & 0.85 & 200 & 0.8 & 0.1 & 40 & 2.5 \\ 38 & 0.8 & 120 & 0.85 & 0.3 & 25 & 3.5 \\ 27 & 0.9 & 95 & 0 & 0.1 & 30 & 4.0 \\ 30 & 0.75 & 180 & 0.95 & 0.2 & 85 & 3.0 \end{pmatrix} \tag{8}$$

Secondly, yielded normalized quality criteria matrix M_S' is shown in the equation (9) according to the step two of the algorithm.

$$M_S' = \begin{pmatrix} 0 & 0.8165 & 1 & 0.9177 & 1 & 0.8660 & 0 \\ 0.3922 & 0.5774 & 0.4880 & 0.9459 & 0 & 1 & 0.8165 \\ 1 & 1 & 0 & 0 & 1 & 0.9574 & 1 \\ 0.8771 & 0 & 0.8997 & 1 & 0.7071 & 0 & 0.5774 \end{pmatrix} \tag{9}$$

Service requester's preference to their objective web services is represented by a quality criteria weight array, which is specified as W={0.35,0.2,0.1,0.05,0.05,0.1,0.15}.

Thirdly, calculating each selected service's synthetic QoS value is performed by the step three of the algorithm. The computational QoS value of each service is formed a QoS mark vector V={0.4458,0.5713,0.8457,0.5689}. Therefore, the descending web service list of QoS mark is $V(WS_3)$> $V(WS_2)$> $V(WS_4)$> $V(BTC)$. Finally, the ranked service set S_R={WS_3, WS_2, WS_4, *BTC*} is generated by comprehensive QoS value calculation and returned to the service requester.

7. Conclusion

In this paper, we have discussed and proposed an approach on how to be able to efficiently select web services with similar functionalities. We firstly gave an overall framework for service selection and ranking based on web service description model *WSDM-Q*. Subsequently, we have given service selection algorithm satisfying for user's basic QoS requirements, service ranking algorithm for normalizing and calculating comprehensive QoS values of all candidate services. Finally, we gave a simple but efficient quality updating mechanism in terms of service requesters' feedback information. Extensible experimental results demonstrate that the proposed framework *WSSR-Q* can fulfill service requesters' non-functional requirements and achieve better web service selection effectiveness.

Acknowledgements

This work is supported by the National High-Tech Research and Development Plan of China under Grant No. 2008AA04Z106, the NSFC under Grant No. 70771077, and the Project of Science and Technology Commission of Shanghai Municipality under Grant No. 08DZ1122300.

References

[1] K. Yue, X. Wang and A. Zhou, "Underlying techniques for web services: A Survey," *Journal of Software*, vol. 15, no. 3, 2004, pp.428-442.

[2] M. Paolucci, T. Kawamura, T. Payne and K. Sycara, "Semantic matching of web services capabilities," *In Proc. of the 1st Intl. Semantic Web Conference*, 2002, pp.333-347.

[3] Z. Xu, P. Martin, W. Powley and F. Zulkernine, "Reputation-enhanced QoS-based web services discovery," *In Proc. of the IEEE Intl. Conf. on Web services*, 2007, pp.249-256.

[4] Y. Liu, A. Ngu and L. Zeng, "QoS computation and policing in dynamic web service selection," *In Proc. of the*

13th Intl. Conf. on World Wide Web, New York: ACM Press, 2004, pp.66-73.

[5] X. Wang, T. Vitvar, M. Kerrigan and I. Toma, “A QoS-aware selection model for semantic web services,” *In Proc. of the 4th Intl. Conf. on Service-Oriented Computing*, 2006, pp.390-401.

[6] D. Roman, U. Keller, H. Lausen, et al, “Web service modeling ontology,” *Applied Ontology*, vol. 1, no. 1, 2005, pp.77-106.

[7] V. Diamadopoulou, C. Makris, Y. Panagis and E. Sakkopoulos, “Techniques to support web service selection and consumption with QoS characteristics”. *Journal of Network and Computer Applications*, vol. 31, 2008, pp. 108-130.

[8] S. Deng, J. Yin, Y. Li, J. Wu and Z. Wu, “A method of semantic web service discovery based on bipartite graph matching,” *Chinese Journal of Computers*, vol. 31, no. 8, 2008, pp.1364-1375.