An Extensible Computing Model for Reputation Evaluation Based on Objective and Automatic Feedbacks[1]

Guang Yang, Hao-peng Chen
School of Software, Shanghai Jiao Tong University, P.R.China, 200240
allenmacyoung@gmail.com, chen-hp@sjtu.edu.cn

Abstract

Service consumer ranking are used by many existing computing models for reputation. Although a variety of methods are proposed, the subjectivity and scarcity of service consumer ranking can hardly be overcome. Moreover, service consumers cannot find service providers with high reputations as well as satisfying their QoS constraints. In order to solve these problems, we proposed an extensible computing model which uses objective and automatic feedbacks to evaluate reputation with the support of measurable QoS constrains. As a typical QoS constraint, performance is discussed in detail. We also talked about invocation fraud issue and presented some general ideas to solve this problem.

1. Introduction

Service computing has been an area of intense activity in recent years. Along with the rapid increase of Web Services on the Internet as well as SOA systems, how to find suitable services has become a more and more important problem and challenging task. In such a scenario, Quality of Service (QoS) serves as a benchmark to differentiate service providers and comprises of techniques that aim to bring a balance between the needs of the service consumers and those of the service providers while being constrained by the limited network and server resources [1].

Many researchers have been trying to define, classify and unify QoS attributes, thus propose an accurate, self-contained QoS model. [2] points out that QoS attributes should contain availability, response time, throughput and security etc; [1] defines QoS attributes to be availability, accessibility, integrity, performance, reliability, regulatory and security; while in [3], QoS attributes is concluded as performance, reliability, scalability, capacity, robustness, exception handling, accuracy, integrity, accessibility, availability, interoperability, security and network-related QoS requirements. Currently expressed as the average user ranking, reputation is a widely acknowledged concept to reflect these non-functional QoS attributes [4, 5].

However, these QoS attributes can be further classified into 2 categories according to their measurability. Some attributes like performance and accessibility can be computed quantificationally while others can hardly be measured accurately. Therefore, we should take these measurable attributes into separate consideration, which leads to a coarse-grained QoS attributes classification used in this paper:

- *Measurable QoS attributes*: every measurable QoS attribute is treated individual attribute.
- *Reputation*: represents the general quality of all the immeasurable Qos attributes as a whole.

So far, most of the existing reputation evaluation models all recommend services according to their reputations only. Typically, reputation is calculated based on the service consumer ranking. However, no matter what computing model it uses, this kind of calculation usually cannot produce good results because of the two assumptions it makes:

1. Service consumers will rate service providers. Ranking is a manual activity which can hardly be performed automatically. One cannot assume a reasonable percentage of consumers would like to rate service providers per service invocation.
2. Ranking itself is a subjective activity. Although in order to get more reliable reputation evaluation, various computing models have been proposed to deal with unfair ranking and prejudicial feedbacks, the results are far from satisfaction.

[1] This paper is supported by the National High-Tech Research Development Program of China (863 program) under Grant No. 2007AA01Z139.

978-0-7695-3273-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ALPIT.2008.20

Furthermore, these models are not able to handle measurable QoS attributes properly so that service consumers cannot find services with high reputations as well as satisfying their QoS constraints. E.g., a service consumer cannot get a recommendation of high reputation services whose response time is less than 400ms.

In order to overcome these weaknesses, we proposed an extensible computing model for reputation evaluation which not only use reputation but also support measurable QoS constraints in its recommendation procedure. Moreover, all the computations are based on the real-time service running status fed back by service consumers rather than subjective user ranking. Notice that these feedbacks are sent automatically with no need of human activities.

Practically, this model will be implemented on service registries such as QoS-aware UDDIs and will involve service consumers as well. Suppose we have a set of service providers which provide services with similar functional requirements, the following algorithm will return a set of services ordered descendingly by their reputations whose measurable QoS attributes can satisfy the given constraints.

$$recommend(constraints)$$

The given constraints can contain any measurable QoS attributes or even a combination of them. In this paper, we only discuss performance. In order to support other type of constraints, one can extend our computing model to add one's own mechanism.

The rest of this paper is organized as follows. In Section 2, we discuss other work in this area. Section 3 makes some assumptions and presents the basic architecture of our computing model. Section 4 dives into the computing model from two aspects: performance and reputation. The entire recommendation algorithm will be shown and we will discuss rectifying invocation fraud as well. Section 5 provides some simulations to demonstrate our method. In the final section, we conclude this paper and discuss some future works.

2. Related Work

In order to obtain more reliable service reputation, many computing models have been proposed. Some researchers focus on performance and security and have proposed some different evalutation model [6, 7]. Some commit themselves to service discovering mechanism and have built some real systems, Among them, [8] has presented the design and implementation of a service discovering mechanism based on QoS; [9] has built a QoS-enabled service registry in a SOA architecture, which extends the existing UDDI standard to provide search capability according to the consumers' QoS criteria; [10] has implemented a reputation-enhanced service discovering algorithm, which is based on the consumer's performance feedbacks to compute QoS. Besides, authors of [11] have proposed an evaluation framework that includes a flexible quality meta-model for formalizing customer and provider views of quality, and a decisional model defining a systematic approach for comparing offered and requested quality of services. While [12] has introduced a new QoS-based semantic web service selection and ranking solution with the application of a trust and reputation management method.

All the papers introduced above are intended to compute service reputation using service consumers' ranking. But as mentioned in Section 1, ranking is a subjective behavior, which may be adulterated with prejudice or even malignity. In order to solve this problem, researchers have mainly proposed two different kinds of solutions:

- One focuses on more precise definition of service reputation.
- The other prefers using algorithms to reduce or even eliminate the unfaithful ranking.

For the first opinion, author of [13] has defined service reputation as a function of user ranking, compliance and verity, in which compliance is described as the service performance recorded by the reputation evaluation system and verity represents the historical QoS values.

For the second opinion, [14, 15] use Bayesian system to filter Unfair ranking. But these methods can hardly ensure the integrality and accuracy of the consumers' feedback. Consumers can differ from each other in many ways. Their ranking would also have different quality. Moreover, even unfair ranking should be considered separately according to if it's just prejudice or malignity. In order to solve this problem, some have proposed a new trust model and implemented a probabilistic method to rectify the prejudice [16]. Some other researchers believe that personal requirements should be considered in reputation evaluation and have proposed their own model [17, 18].

These related work all consider service reputation as the general representation of all QoS attributes and mainly use subjective consumer ranking to evaluate reputation. This paper defines reputation in a more reasonable way and supports measurable Qos constraints in service recommendation. Moreover, reputation is computed using objective consumer feedback, thus can avoid unfair ranking issue.

3. Assumptions and basic architecture

Making full use of service consumer feedbacks, our computing model is likely to reside on a service registry. In order to further discuss this model and for the sake of convenience, we first declare the following assumptions:

- All service consumers discover services through service registry. So we don't consider any

provider-consumer relationships without any service registry.

- All service providers publish their services through a service registry. So we don't consider any providers who publish services in their own way.
- There is only one service registry. So we don't consider any distributed problems like feedback synchronization.
- Every service consumer or provider has a universal unique ID (UUID).
- There has already been an efficient solution to find service providers according to functional requirements.
- Only consider response time for performance.

Obviously, these assumptions are reasonable and easy to extend. We will see that they are not deterministic factors of our computing model. With these assumptions, we can focus on the QoS aspect of service computing.

Fig 1 illustrates our computing model on a high level. All the solid lines in this figure are already implemented in modern service computing environment, while all three dashed lines are proposed by our model. A typical procedure of our computing model would be as follows:

1. Service consumers send real-time services running status to service registry.
2. Service registry uses our computing model to calculate the measurable QoS attributes and reputation of each service.
3. When some service consumer asks service registry to recommend services, service registry filters service with the given QoS constraints first and then responses the request with the final recommendation list.

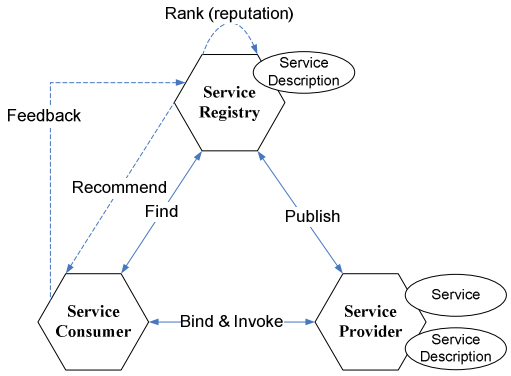

Figure 1. Basic architecture of the computing model

As mentioned in Section 1, we only consider performance for measurable QoS attributes in this paper. So the algorithm can be abstracted as follows:

$$recommend(\langle performance_{max}, sc\rangle)$$

In this algorithm, $performance_{max}$ is the upper bound of the performance constraint and sc is a data structure which contains service consumer information such as UUID and IP address. Notice that we will compute *consumer-aware performance* rather than merely service provider's runtime performance. This is because that consumer-aware performance is highly depended on the geographical locations of the consumers and providers as well as the topological structure and bandwidth of the networks between the two. Service consumers at different locations or in network segments will likely have a totally different performance experience for the same service provider.

One can easily extend our model to support other measurable QoS attributes. The only work is to find a way of computing these attributes and adapt it into our model. The main recommendation process should remain the same.

4. Computing model

As mentioned above, our computing model considers both performance and reputation. Both of them are dynamic QoS attributes and are impossible to measure accurately. Therefore, we adopt a statistical and sociological method to compute these two attributes. We will introduce performance computing first and then go further to the reputation ranking algorithm. After that, we will show the entire recommendation model. At last, we will discuss how to rectify invocation fraud.

Before starting to introduce our computing model, we first define two variables that will be used as the basic data. Both of them can be easily obtained by service registry.

- S_{sp}: A set of all the service providers which have published services in the registry.
- S_{sc}: A set of all the service consumers which have discovered services through the registry.

4.1. Consumer-ware performance

In our model, service consumers will monitor the real-time status of the services they are using and send the performance status back to service registry. *Performance*, to put it simply, is how quickly the system can respond to a given logical operation from a given individual user. *Response time* is a measure of the amount of time the system consumes while processing a user request, which is made up of three parts: *latency*, which is the amount of time spent processing overhead just to get to the point of carrying out a service; *wait time*, which is the time spent waiting for the service, or, while the service is executing, the time spent waiting for resources; and *execution time*, which is the time needed to process the request when no waiting is involved.

Many researchers prefer to treat latency, wait time and execution time individually and have proposed many algorithms to compute them. This is very important for

performance improving because it can tell which part is the bottleneck. But for consumer side performance monitoring, we should consider response time as a whole. Because consumers don't care about the exact value of these three parts but only focus on the entire response time of their service invocations, which is defined as *consumer-aware response time* in this paper.

Since we take only consumer-aware response time in consideration, the feedback should be a six-dimensional vector fb in which sp is a data structure containing service provider information, s is a data structure containing service information published by service provider, sc is a data structure containing service consumer information, $\bar{t}$ is the average consumer-aware response time, f is the feedback frequency (that is why average response time is used) and ts is a time stamp.

$$fb = \langle sp, s, sc, \bar{t}, f, ts \rangle \tag{1}$$

Feedback frequency can be specified in a variety of forms, such as time (e.g. once per five minutes) or invocation count (e.g. once per thousand service invocation). Here we select invocation count because it is convenient for performance computing. Another reason will be explained in the next section. With all the feedbacks aggregating together, we will obtain a set of feedbacks S_{fb}:

$$S_{fb} = \{fb\} \tag{2}$$

Next, we define a function $netseg(sc)$, which returns a network segment which contains the given service consumer. Suppose $netseg(sc)$ can map every single service consumer into a reasonable network segment, we can declare that all the service consumers in the same network segment should have the same network condition. Therefore, we can classify all the feedbacks according to its reporter's network segment. For a certain service consumer and provider, we can obtain all the feedbacks from the same network segment.

$$S_{fb}(sc, sp) = \left\{ fb \middle| \begin{matrix} fb \in S_{fb} \\ netseg(fb.sc) = netseg(sc) \\ fb.sp = sp \end{matrix} \right\} \tag{3}$$

$$S_{fb} = \bigcup_{i=1}^{|S_{sc}|} \bigcup_{j=1}^{|S_{sp}|} S_{fb}(sc_i, sp_j) \tag{4}$$

Then for a certain service consumer, we can compute its global average consumer-aware response time for each service provider using equation 5.

$$S_{\bar{T}}(sc) = \left\{ \langle sp, \bar{T} \rangle \middle| \bar{T} = \frac{\sum_{i=1}^{|S_{fb}(sc,sp)|} fb.\bar{t} \cdot fb.f}{\sum_{i=1}^{|S_{fb}(sc,sp)|} fb.f} \right\} \tag{5}$$

Notice that for a certain service consumer we only take feedbacks from the same network segment of the consumer into consideration. This is because feedbacks from other network segments can make against the computation of consumer-aware performance for the service consumer. E.g. for a service provider in America, ten American consumers get a 200ms average performance while ten Chinese consumers have to wait 4000ms to get the invocation response. Now another American consumer asks for recommendation, the average performance for this consumer is supposed to be 200ms. But if we take Chinese consumers into account, the performance will become 2100ms, which makes a big difference.

Finally, we add an additional historical constraint to the above equation, declaring that only feedbacks in 24 hours will be considered. This is based on the fact that performance is a critical QoS attributes that good service providers should not leave it slow for a relatively long period. This way, we can eliminate some unwanted historical cumulative feedbacks and take advantage of average value at the same time.

Notice that $netseg(sc)$ should be implemented in network layer and it's out of our concern. We provide a flexible mechanism here that other researchers can implement their own network segment solution and easily adapted into our computing model.

4.2. Reputation

As mentioned in Section 1, traditional computing models for reputation are based on the user ranking and have several weaknesses. In order to solve the problems, our model makes use of a sociological theory to eliminate user ranking and introduce two new factors: service consumer count and service invocation count. Sociologically, services with higher reputations should have more consumers and service invocations, vice versa. Therefore, we can compute a service's reputation according to the amount of its consumers and the amount of its service invocations. Notice that these two factors can be obtained objectively and automatically while user ranking is much more subjective and need manual work.

As shown in equation 1, consumer feedback contains feedback frequency in the form of invocation count, thus we can obtain the daily service consumer count n_{sc} and service invocation count n_{si} for each service using equation 6 and 7. That is why we choose invocation count as the form of feedback frequency.

$$n_{sc}(s) = \left|\left\{fb.sc \middle| \begin{matrix} fb \in S_{fb} \\ fb.s = s \end{matrix}\right\}\right| \quad (6)$$

$$n_{si}(s) = \sum_{i=1}^{\left|\left\{fb \middle| \begin{matrix} fb \in S_{fb} \\ fb.s = s \end{matrix}\right\}\right|} fb_i.f \quad (7)$$

Then we use geometric mean of the above two factors to express the daily traffic of a service:

$$traffic(s) = \sqrt{n_{sc}(s) \cdot n_{si}(s)} \quad (8)$$

Furthermore, we aggregate the historical daily traffic of a service for a certain period. Thereby, we can obtain the daily traffic trend for every service. We will use these trends to compute the reputation of every service.

When computing reputation, two properties of the trend should be considered: average daily traffic and trend consistency. For average daily traffic, we simple calculate the average of all the historical daily traffic ($\overline{traffic(s)}$) and we use standard deviation of the trend sample space ($\sigma(s)$) to express trend consistency. Obviously, reputation should be an increasing function of $\overline{traffic(s)}$ and a decreasing function of $\sigma(s)$. Thus we use the following equation to calculate reputation:

$$reputation(s) = \begin{cases} 0 & if\ \overline{traffic}(s) < TV_{traffic} \\ \sqrt{\dfrac{\overline{traffic(s)}}{\ln \sigma(s)}} & if\ \overline{traffic}(s) \geq TV_{traffic} \end{cases} \quad (9)$$

$TV_{traffic}$ is a configurable constant serving as the threshold value of average daily traffic. In other words, only services whose average daily traffic reaches a certain level will have reputations. This is necessary because otherwise services with every low but nearly changeless traffic trend will get an extremely high reputation.

Furthermore, we use $ln\,\sigma(s)$ to reduce the impact of trend consistency. This is because in actual environment, $\sigma(s)$ is likely to be very large, which will lead to an unwanted result that $\Delta reputation$ is relatively small.

4.3. Recommendation algorithm

Based on the computing model introduced in the above two sections, we can now define the recommendation algorithm as follows:

Line 1 uses equation 5 to calculate consumer-aware performance of each service. Line 2 creates a set whose element is a two-dimensional vector containing a service and the corresponding reputation. Line 3 to line 6 iterates every service, eliminates the ones which cannot satisfy performance requirement and computes reputations for the rest using equation 9. Line 7 sorts all the valid services in descending order according to their reputations. The last line returns the result.

```
recommend(performance_max, sc)
1  set_s ← S_T̄(sc)
2  set_reputaion ← {new set of ⟨s, reputation⟩}
3  for each item in set_s
4      do if item.T̄ > performance_max
5          then delete item from set_s
6          else set_reputaion ←
                   ⟨item.s, reputation(item.s)⟩
7  sort set_reputation accroding to reputation
8  return set_reputation
```

Figure 2. Recommendation algorithm

4.4. Rectify invocation fraud

One problem of the above computing model is invocation fraud. Because the entire computing model is based on service consumers' feedbacks, a service provider can simply set up several fake service consumers which continually invoke the services provided by itself. This way, the reputations of the services provided by this service provider could go much higher than their real values.

Similar problem named *click fraud* has existed in pay-per-click online advertising systems like Google for a long time. These systems allow everyone who has a website (e.g. a blog owner) choose advertisements from the system and publish them on their own website. When someone clicks the ad, the system will charge the advertiser some money and pay a small part of that money to the website owner. While in these systems, website owner can earn more money by performing click fraud, in our reputation evaluation model, invocation fraud is performed by service providers to increase their reputation. But the general idea of solving these problems is the same: eliminate fraudulent behaviors or reduce the effect of them.

A variety of proposals for reducing click fraud have surfaced. Some ideas can be adapted in to our model to solve the invocation fraud issue.

- *Invocation per IP*: the invocation count from very single IP.
- *Invocation frequency*: the time between two invocations from the same IP.

Based on the large feedback sample space we collect, the above two factors should follow Possion distribution. For instance, suppose a service's average invocation per IP in a day is 500 and average time between two invocations is 2 minutes, and we find that one consumer call this service 300,000 a day with a frequency of 15 seconds, then these invocations are likely to be fraudulent.

Since we have traffic trends, *invocation per timeframe* can be used as well. If the invocation count of a service increased tremendously in a very short time, then there is a potential that invocation fraud has just happened.

Besides these ideas which come from the click fraud research area, we can further use a unique factor of SOA system: *service substitution*. When a consumer asks service registry to recommend some services, it will choose one to use from the response list. If that service is not good enough, then the consumer is likely to substitute it with another one. We can make consumers send these substitution behaviors back to service registry and build a machining learning algorithm using these feedbacks to refine the reputation. Notice that since our recommending algorithm supports measurable QoS constraints, all the recommended services can satisfy the consumers' QoS requirements, so we can consider that many substitutions on a single service indicates that the service may have an unreasonable high reputation.

5. Simulation

A system prototype has been built for illustrating our computing model. In the prototype, we have one service registry, three services (s_1, s_2, s_3) and hundreds of service consumers from three different network segments (ns_1, ns_2, ns_3). We will begin with performance and then go on to reputation.

Some typical service consumer feedbacks are shown in Table 1. For the sake of convenience, we only show average response time (in millisecond) and feedback frequency here.

Table 1. Simplified service consumer feedbacks

Network segment	s_1		s_2		s_3	
	$\bar{t}$	f	$\bar{t}$	f	$\bar{t}$	f
ns_1	492	1000	177	1000	429	1000
	474	800	183	800	437	800
	…	…	…	…	…	…
	482	1000	212	1000	418	1000
ns_2	206	600	516	600	356	600
	206	100	495	100	361	100
	…	…	…	…	…	…
	215	2000	518	2000	387	2000
ns_3	323	500	388	500	274	500
	326	1000	417	1000	293	1000
	…	…	…	…	…	…
	339	1000	384	1000	262	1000

According to equation 5, we can compute the average consumer-aware performance of each service for each network segment. Table 2 shows the result and we can see that service consumers from different network segments indeed feel distinct performance.

Table 2. Average consumer-aware performance

Network segment	s_1	s_2	s_3
ns_1	484	191	421
ns_2	221	518	368
ns_3	330	404	273

For the computation of reputation, we first calculate the daily traffic for each service in the past one hundred days using equation 8. Part of the calculating result is shown in Table 3 as follows.

Table 3. Historical daily traffic

Time(day)	s_1	s_2	s_3
0	548	471	513
10	578	389	585
20	601	399	488
30	615	441	483
40	619	481	520
50	615	508	516
60	605	522	499
70	593	532	537
80	587	547	563
90	594	566	405
100	625	576	508

Having this historical daily traffic, we can obtain the traffic trends.

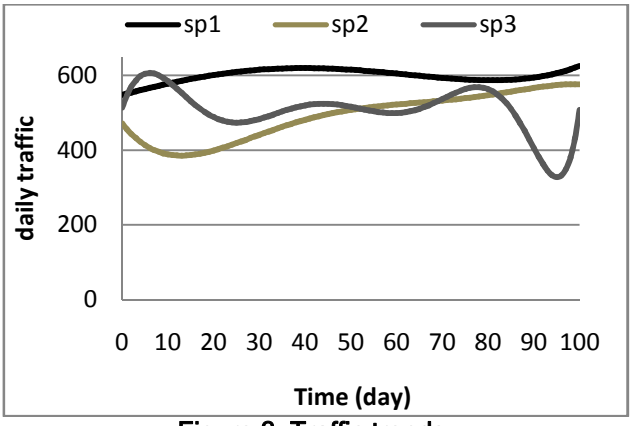

Figure 3. Traffic trends

Now according to equation 9 and setting $TV_{traffic}$ to 450, we can compute the reputations of all three services. Table 4 shows the final result.

Table 4. $\overline{traffic}$, σ and reputation

Service provider	$\overline{traffic}$	σ	$reputation$
s_1	599.62	16.54	14.62
s_2	490.01	62.19	10.89
s_3	506.81	59.99	11.13

With Table 2 and 4, we can get the recommendation list for different service consumers. Table 5 shows some typical scenarios.

Table 5. Recommendation result

Service consumer	$performance_{max}$	Recommendation result
Form ns_1	500	$\langle s_1, s_3, s_2 \rangle$
	430	$\langle s_3, s_2 \rangle$
Form ns_2	370	$\langle s_1, s_3 \rangle$
	200	$\emptyset$
Form ns_3	400	$\langle s_1, s_3 \rangle$
	300	$\langle s_3 \rangle$

6. Conclusion

Most of the existing computing models for reputation are all based on service consumer ranking. Basically they cannot produce satisfying results because ranking are quite subjective and service consumers do not always intend to send feedbacks, not to mention prejudicial ranking. In order to solve the problem, we proposed an extensible computing model based on objective and automatic feedbacks using some simple statistical and sociological theories. Moreover, measurable QoS attributes has been taken into consideration so that service consumers can find service providers not only have high reputations but also are able to satisfy their QoS constraints, especially performance.

This paper is supposed to be an initial research on computing QoS using objective and automatic feedbacks. The computing model proposed here needs improvement as well as extension in order to support other measurable QoS attributes. Rectification of invocation fraud also needs further study. Besides, since we have the traffic trends, we plan to use time series theory to implement fault prediction, with which we can not only get the reputations of service providers but also detect which providers are likely to fail at a certain point in the near future.

6. References

[1] A. Mani and A. Nagarajan, "Understanding Quality of Service for Web Services", http://www-106.ibm.com/developerworks/library/ws-quality.html, January 2002.

[2] Menasce D.A., "QoS Issues in Web Services", *IEEE Internet Computing*, November-December 2002, vol. 6, pp. 72-75.

[3] K. Lee, J. Jeon, W. Lee, S.-H. Jeong, and S.-W. Park, "QoS for Web Services: Requirements and Possible Approaches", http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/ W3C Working Group, 2003.

[4] Chang E., Dillon T.S., and Hussain F.K., "Trust and reputation relationships in service-oriented environments". *The Third International Conference on Information Technology and Applications (ICITA 2005)*, IEEE Computer Society, Sydney, Australia, July 2005, vol. 1, pp.4-14.

[5] Haiyan zhao, Hong Mei. A Service-Oriented Trust Management Model on Application Server, The IEEE International Conference on Web Services(ICWS 2006), IEEE Computer Society, Chicago, USA, September 2006, pp. 170-177

[6] Chen Shiping, Zic John, Tang Kezhe, Levy David. Performance Evaluation and Modeling of Web Services Security, The IEEE International Conference on Web Services(ICWS 2007), IEEE Computer Society, 9-13 July 2007, pp. 431-438

[7] Shiping Chen, Bo Yan, Zic J., Ren Liu, Ng A. Evaluation and Modeling of Web Services Performance, The IEEE International Conference on Web Services(ICWS 2006), IEEE Computer Society, Chicago, USA, September 2006, pp. 437-444

[8] Makripoulias Y., Makris C., Panagis Y., Sakkopoulos E., Adamopoulou P., Tsakalidis A.. Web Service discovery based on Quality of Service, The IEEE International Conference on Computer Systems and Applications 2006, IEEE Computer Society, March 8, 2006, pp. 196-199

[9] Anup Kumar, El-Geniedy A., Sanjuli Agarwal. A generalized framework for providing QoS based registry in service oriented architecture, Proceeding of SCC 2005 Conference(SCC 2005), IEEE Computer Society, 11-15 July 2005, pp. 295-301 vol. 1.

[10] Xu Ziqiang, Martin Patrick, Powley Wendy, Zulkernine Farhana. Reputation-Enhanced QoS-based Web Services Discovery, The IEEE International Conference on Web Services(ICWS 2007), IEEE Computer Society, 9-13 July 2007, pp. 249-256

[11] Casola V., Fasolino A. R, Mazzocca N., Tramontana P.. A policy-based evaluation framework for Quality and Security in Service Oriented Architectures, The IEEE International Conference on Web Services(ICWS 2007), IEEE Computer Society, 9-13 July 2007, pp. 1181-1190

[12] Le-Hung Vu, Manfred Hauswirth, Karl Aberer. QoS-Based Service Selection and Ranking with Trust and Reputation Management, CoopIS/DOA/ODBASE 2005, LNCS 3760, pp. 466-483

[13] S. Kalepu, S. Krishnaswamy, Seng Wai Loke, Reputation = f(user ranking, compliance, verity), IEEE International Conference on Web Services 2004 Proceedings(ICWS 2004), 6-9 July 2004, pp. 200 – 207.

[14] Withby, A., A. Jøsang, and J. Indulska, Filtering Out Unfair Ratings in Bayesian Reputation Systems. The Icfain Journal of Management Research. 2005, volume 4, issue 2, pp. 48-64.
[15] Wu Guoquan, Wei Jun, Qiao Xiaoqiang, Li Lei. A Bayesian Network Based Qos Assessment Model For Web Services, Proceeding of SCC2007 Conference(SCC 2007), IEEE Computer Society, Salt Lake, Utah, USA, July 2007, pp. 498-505
[16] Yanzhen Zou, Liang Gu, Ge Li, Bing Xie, Hong Mei, Rectifying Prejudicial Feedback Ratings in Reputation based Trust Management, Proceeding of SCC 2007 Conference(SCC 2007), IEEE Computer Society, Salt Lake, Utah, USA, July 2007, pp. 530-535
[17] Shao Lingshuang, Zhang Jing, Wei Yong, Zhao Junfeng, Xie Bing, Mei Hong. Personalized QoS Prediction for Web Services via Collaborative Filtering, The IEEE International Conference on Web Services(ICWS 2007), IEEE Computer Society, 9-13 July 2007, pp. 439-446
[18] Hung Jeng-Shin, Liu Alan. Using Personal Ontology in Evaluating Service Quality, Proceeding of SCC 2007 Conference(SCC 2007), IEEE Computer Society, Salt Lake, Utah, USA, July 2007, pp. 332-339