International Journal of Grid and Distributed Computing
Vol.7, No.1 (2014), pp.1-12
http://dx.doi.org/10.14257/ijgdc.2014.7.1.01

An Approach to Web Service Dynamic Replacement

Yan Gong[1], Lin Huang[1,2], Fan Jiang[1] and Ke Han[1]

[1]*Chinese Electronic Equipment System Engineering Corporation, Beijing, China*
[2]*State Key Laboratory of Networking and Switching Technology,*
Beijing University of Posts and Telecommunications, Beijing, China
gongyan@bupt.edu.cn, huanglin1204@gmail.com, eddi.ef@163.com,
kehan@yahoo.com.cn

Abstract

Service-oriented software is composed of a combination of services. Because of the dynamic nature of the Web service environment, the QoS often fluctuates with inherent uncertainty and therefore needs to be dynamically replaced. Based on the changes in the QoS of the software itself and users' requirements for the software, this paper proposes a dynamic replacement approach based on the cloud model. It adopts the cloud model to transform the quantitative QoS to qualitative QoS for the uncertainty computation and judges the uncertainty level by setting the corresponding service QoS parameters. When a member service with unstable QoS is found, it will be replaced with the candidate service with the same or better QoS to stabilize the QoS of the entire software. Both theoretical analysis and experiment results have proven the feasibility and effectiveness of this approach.

Keywords: *Web Service; Cloud Model; QoS; Service Oriented Architecture (SOA)*

1. Introduction

With the development of Internet technology, the Service Oriented Architecture (SOA), as a new distributed computing model, has become more and more advanced. It features strong reusability and adaptability and is evolutionary. Typically, the software based on SOA consists of a large number of Web services. Web service is the core technology to implement SOA and the basic functional entity to build service-oriented software. Running in a dynamic environment, the Web service is faced with various changes [1-3]. The QoS of the existing Web services may be unstable, but new Web services will continuously appear with presumably better QoS [4, 5]. Therefore, after composing the Web services, continuous maintenance should also be conducted to ensure that the QoS of the service composition will continually meet the requirements [6, 7]. The heterogeneous, dynamic and changeable characteristics of the Web service make the SOA-based software centered on it easier to achieve dynamic replacement.

Dynamic substitution is not a new notation and many solutions have already been proposed to deal with the distributed systems and software architectures [8-11]. These works focus on the substitution towards the function of the software, but few works address the reconfiguration of the QoS. However, dynamic substitution can help the SOA-based software not only to update, modify, add and remove their functions, but also to improve their performance and enhance their reliability and stability at runtime. In this paper, we focus on the nonfunctional substitution of the SOA-based software, for example maintaining and improving the QoS.

ISSN: 2005-4262 IJGDC

The previous concerns about the service QoS tended to focus on the users' changes in their requirements towards it. By determining what service needs to be replaced and by conducting the dynamic replacement, the SOA-based software will re-achieve the users' QoS requirements. However, it should be noted that the QoS value of the Web service that the software is made up of has significant fluctuations, which will impact on the stability of the entire software's QoS value and lead to its failure in meeting the users' needs. Thus, a way to monitor and provide dynamic replacement to the Web services of the software is needed to improve its QoS stability.

This paper presents a service dynamic replacement method based on the cloud model. For QoS value fluctuations of the software's member services, a monitoring module is introduced to do real-time monitoring. It uses backward cloud algorithm of the cloud model to turn quantitative QoS into qualitative QoS and assesses the level of its QoS instability according to the threshold parameter that has been set up in advance. Once a member service is found with great QoS value fluctuation, another service from its candidate list will be selected to replace it, stabilizing the QoS of the software.

The remainder of this paper is organized as follows. Section 2 introduces the related works of service substitution. Section 3 describes the service dynamic substitution based on cloud model. Section 4 shows the experiments and Section 5 concludes the paper.

2. Related Work

Service dynamic replacement is an important research direction of the service computing area, which has attracted the attention of many researchers and has achieved many significant results. Below are some of the representative service replacement methods.

Paper [12] introduces an approach to service substitution based on user preferences over non-functional properties of services. The approach utilizes preference networks for representing and reasoning about preferences over non-functional properties. Paper [13] introduces the QoS maintenance problems of the distributed system based on Web services. Since the Web services constraints may change in the communication or implementation phase, this paper presents a self-healing middleware framework to manage the QoS of the distributed system based on Web services. The framework monitors and analyzes the QoS values of the system and re-configures it. Paper [14] proposes a method to fix the services that do not work. It replaces dysfunctional services with the new ones, and ensures the new services provide the end-to-end QoS requirements of the users. Service substitution based on the functional properties of components has been addressed by many authors [15-17]. These papers focus on the fixing of the software. This paper, however, focuses on the instability of the software's non-functional properties during its running and aims to conduct unstable dynamic replacement to the member services to meet the users' requirements.

Paper [18] puts forwards a dynamic substitution approach of QoS-driven services. It records QoS reference value of each member service in a list, extracts their real time QoS values once over a period of time and compares them with the reference values. If the difference is greater than the given threshold value, the counter of the member service will add one. Once the number of the counter reaches the limit, which means the QoS value of the member service is very unstable, the member service will go to the service list that requires updates and the operation of service replacement will be triggered. This method has the following three disadvantages: first, it cannot effectively detect the uncertainty of QoS, leading to the inaccuracy in selecting the services that need replacement; second, it cannot select the service of stable QoS values from the

candidate list or ensure the success of the service replacement; third, it cannot radically narrow the search space of the service replacement or remove redundant Web services, resulting in a longer time of service replacing. This paper will compare it with the research method we propose through experiments.

3. Web Service Dynamic Replacement (WSDR)

There are two main conditions that will trigger the dynamic service replacement operation during the running of the software. First, when the result from the real-time monitoring that tracks the stability of member services finds the QoS value of a member to be very unstable, it will be replaced with another service with stable QoS value on its candidate list, so the QoS value of the software continues in a stable state. Second, when the QoS value of the current software cannot meet users' new QoS requirements, its member services will also need to be replaced dynamically. The re-configured software, after the service dynamic replacement triggered by the two situations above, will continuously meet users' requirements. This paper focuses on the first situation.

In the service-oriented software, the main operating of the non-functional dynamic replacement is service replacement. The dynamic replacement of non-functional properties does not change the logical structure of the software. It only hopes to stabilize the software by replacing one or more of its member services. The service dynamic replacement process shows in Figure 1. The proposed WSDR approach contains two phases. The first phase is QoS instability detection (Section 3.1), in which we adopt cloud model to transform the quantitative QoS to qualitative QoS for the QoS uncertainty computation. The second phase is service substitution (Section 3.2), which is to find the service that meets the requirements from the candidate services and use it as replacement.

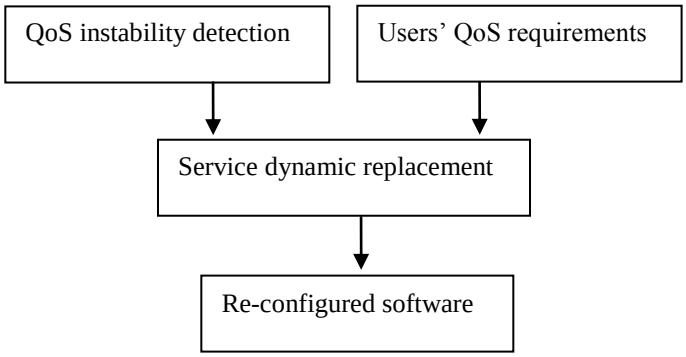

Figure 1. Procedures of WSDR Approach

3.1. QoS instability Detection

Most SOA-based software is composed of a large number of Web services, which have the open, dynamic and changeable features. Web service might show instability of non-functional attributes, affecting the user experience. For example, the Web services, widely distributed in ubiquitous network, may come from different organizations and run on different system platforms. Fluctuation from their location or network is likely to affect the quality of it running. To solve the instability, this paper has designed a module to test the non-functional attributes of the SOA-based software. The module will examine the QoS value of each service in the service-oriented software. When it detects great changes in service QoS value,

it will decide to replace the member service with another one from its candidate list with stable QoS values in order to stabilize the QoS values of the software. We adopt a new perspective to study the problem of instability in the Web service and the core is to use the cloud model to compute QoS uncertainties and to identify the service to be replaced.

To evaluate the degree of the service QoS uncertainty and ensure the reliable service replacement, WSDR adopts cloud model to compute the uncertainty by transforming quantitative QoS values (transaction logs) to qualitative QoS concept (uncertainty level). According to the uncertainty level, a web service with consistently bad QoS can be distinguished from those services with a large QoS variance.

3.1.1. Cloud Model: Cloud model [19] is a model of uncertainty transition between a linguistic term of a qualitative concept and its numerical representation. It can be employed for the uncertainty transition between qualitative concept and quantitative description.

Let U be the set as the universe of discourse, and C a qualitative concept associated with U. The membership degree of quantitative numerical representation x in U to the concept C, $\mu(x) \in [0,1]$, is a random number with a stable tendency, that is as in (1):

$$\mu: U \rightarrow [0,1], \forall x \in U, \; x \rightarrow \mu(x). \tag{1}$$

The distribution of x in the universe of discourse U is called cloud $C(X)$, and x is called a cloud drop. The literature [19], gave many kinds of cloud models, such as normal cloud, γ cloud. Because a lot of uncertainty concepts behave like normal clouds in social and natural phenomena, in our study, we mainly apply normal cloud model. The overall characteristics of cloud model may be reflected by its three numerical characteristics: Expected value (*Ex*), Entropy (*En*) and Hyper-Entropy (*He*). In the discourse universe, *Ex* is the position corresponding to the center of the cloud gravity, whose elements are fully compatible with the linguistic concept; *En* is a measure of the concept coverage, *i.e.*, a measure of the fuzziness, which indicates how many elements could be accepted to the qualitative linguistic concept; and He is a measure of the dispersion on these cloud drops, which can also be considered as the entropy of *En*. Then, the vector $NC = \{Ex, En, He\}$ is called the eigenvector of cloud model.

By using the three numerical characteristics *Ex*, *En*, and *He*, the cloud generator [19] can produce many cloud drops. The transforms between a qualitative concept expressed by the three numerical characteristics of a cloud and its quantitative representation expressed by a set of numerical cloud drops are performed by cloud generators: forward cloud generator and backward cloud generator. In this study, we apply these three numerical characteristics of backward cloud generator (Algorithm 1) to denote the uncertainty of QoS by transforming QoS quantitative values to qualitative concept.

Algorithm 1: Backward cloud generator.

Input: n transactions of a web service, i.e., n cloud drops $\{x_1, x_2, ..., x_n\}$.

Output: the three numerical characteristics *Ex* , *En* , and *He* of the n cloud drops.

1: $Ex = \overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$;

2: $T^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})^2$;

3: $En = \frac{\sqrt{\pi/2}}{N}\sum_{i=1}^{N}|x_i - Ex|$;

4: $He = \sqrt{T^2 - En^2}$.

The working process of algorithm 1 is as follows:

- Line 1: According to x_i, compute the sample mean $\bar{X}$. The expected value of the web service on its QoS can be calculated by $Ex = \bar{X}$;
- Line 2: Compute the sample variance T^2;
- Line 3: Compute estimated value of *En*;
- Line 4: Compute estimated value of *He*.

3.1.2. Application of the Cloud Model: Detection module works as follows: Every once in a certain period of time, it extracts the QoS value of each member service. In order to use cloud model to compute the uncertainty of the Web service QoS, set the parameters λ and h, respectively as the threshold values of *He* and *En*, for evaluating the stability of the Web service. If the monitored Web service satisfies the following conditions:

$$En \le \lambda \text{ and } He \le h \quad (2)$$

It indicates that its service QoS is stable. Otherwise, the QoS of the service is unstable, and of large variational amplitude and poor reliability. And the member service will be added to the list of services that need updating and the service replacement operation will be triggered. Through constant monitoring of the module, member services of unstable QoS values will be found, ensuring the QoS values of the software stay in a stable state.

Algorithm 2: Monitor.

Input: an application $S = \{S_1, S_2, ..., S_n\}$.

Output: needReplaceList with unstable QoS or Null.

1: List substitutionList;

2: SET table *A* with $\{a_1, a_2, ..., a_m\}$ of S_n;

3: SET table *T* with $\{t_1, t_2, ..., t_i\}$ of T_n;

4: Every time unit t_i {

5: FOR j = 0 to Size(A) {

6: Get every QoS of S_n {a_1, a_2, ..., a_m};

7: };

8: };

9: By using *En* and *He* to monitor QoS transactions;

10: SET temp*1* = *En* and temp*2* = *He*;

11: Get every temp*1* and temp*2* of S_n {a_1, a_2, ..., a_m};

12: IF(temp*1* > λ) and (temp*2* > h);

13: SubstitutionList. add(S_n);

14: Return substitutionList.

The working process of algorithm 2 is described as follows:

- Line 1 - Line 2: demonstrate the QoS value list of member services.
- Line 3 - Line 8: gain and store the QoS attribute values of all the member services in each unit of time.
- Line 9 - Line 11: calculate *En* and *He* through the backward cloud algorithm's quantitative - qualitative conversion function of the cloud model.
- Line 12 - Line 14: compare the *En* and *He* with the given thresholds λ and h. If the comparison result is greater than a defined given threshold, the service will be added to the list of services need to be replaced.

Complexity of each test of the algorithm depends on its for-loop. Supposing a software has n service members and each member of the service has m QoS attributes, to go through all of the member services and the corresponding attributes, perform for-loop for $n * m$ times, and therefore its complexity is O ($n * m$).

3.2. Substitution Algorithm

The input of replacement algorithm is composed of a set of services with unstable QoS values. The algorithm is intended to replace the member services of the collection. It will select the suitable services from the candidate list serve the same function to take the place.

Algorithm 3: Service Substitution.

Input: a request of a new QoS $S'_n = \{a'_1, a'_2, \ldots, a'_m\}$, a set M_i of all Candidate services of S_i, substitutionList with unstable QoS.

Output: The new application S' or Null.

1: IF(substitutionList! = Null) {

2: For i = 0 to Size(substitutionList) {

3: IF(exist $S_{ji} \in M_i$ and the QoS of S_{ji}

satisfy ($a'_1, a'_2, \ldots, a'_m$) and $En \leq \lambda$, $He \leq h$) {

4: replace S_j with S_{ji}; Return S';

5: } ELSE{

6: Return Null;

7: };

8: }.

- Line 1 - Line 2: The input of replacement algorithm is composed of a set of services with unstable QoS values.
- Line 3 - Line 8: Find the candidate services satisfying requirements and make replacement.

4. Experiment

In order to verify the feasibility and effectiveness of the cloud model in service dynamic substitution, we conducted experiments.

All the experiments are conducted on the same computer with Pentium 3.0 GHz processor, 4.0GB of RAM, Windows. The experiments are based on the following assumptions: a software contains six member services; all member services always have suitable candidate services that can be selected; All services have three QoS attribute values (execution time, reputation, execution cost); the entropy and hyper entropy threshold values of the QoS uncertainty calculation are set as λ = 3.2, h = 5.4; the execution time for service is monitored and service dynamic replacement will be conducted based on the monitoring results. Monitoring information of each service's execution time is shown in Figure 2.

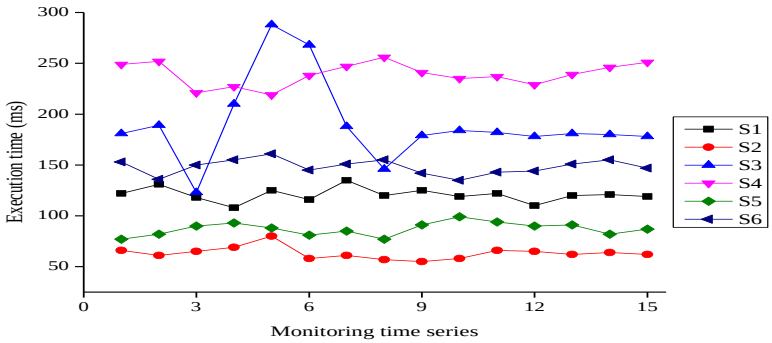

Figure 2. Monitoring Information

As can be seen from Figure 2, when some member services of the software show fluctuation in QoS, such as the service S3 shows great fluctuations in service execution time in the middle stage, our approach will replace them dynamically to stabilize the QoS values of the software and to meet users' expectations.

As can be seen from Figure 3, regardless of how many candidate services there are, the success rate of the WSDR is significantly better than the QDSDS. The average success rate of WSDR is up to 97.5% while with QDSDS, it is only 83.1%. The reason for the WSDR's significant success rate lies in the cloud-based QoS model used in the service replacement, which excludes the candidate services of great QoS fluctuations and adopts those with stable QoS values. Thus the success rate of the service replacement is improved.

Figure 4 shows the contrast results between WSDR and QDSDS on time consumption and other aspects. As can be seen from the experimental results, despite the number of the candidate services, WSDR costs less time in every comparison and can significantly reduce the time cost in service replacement. It is effective to meet real-time requirements of the users' towards the service.

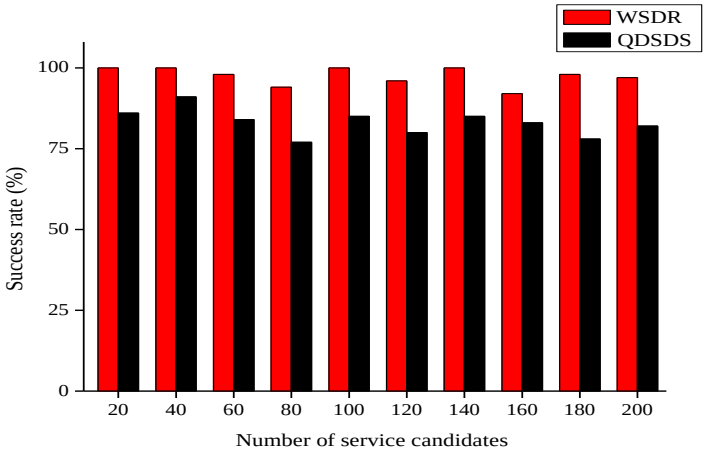

Figure 3. Success Rate

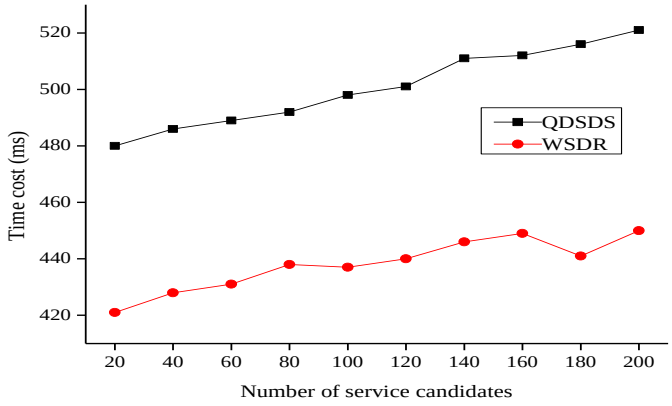

Figure 4. Time Cost

5. Conclusion

Performance of web service may fluctuate due to the dynamic Internet environment, which makes the QoS inherently uncertain. It is necessary to reconfigure the web service to enable its QoS values to meet users' demands. In this paper, we propose an efficient and effective dynamic substitute approach based on cloud model. Our approach employs cloud model to compute the QoS uncertainty to determine dynamic substitute targets. By targeting substitutions, the reconfigured web service will better satisfy users' requirements. The results of some initial evaluation are also described in the paper.

In our future work, we will continue to study more efficient web service substitution approaches (*e.g.*, how to set the threshold of *En* and *He* accordingly). We are currently constructing a realistic test bed and developing a practical middleware for service substitution. Our goal is to help users find the unstable member services and replace them dynamically with the candidate services.

Acknowledgements

This work is supported by the China Postdoctoral Science Foundation funded project (No.2012M521839).

References

[1] K. Mahbub and A. Zisman, "Replacement Policies for Service-based Systems", Proceedings of the 7th International Conference on Service Oriented Computing, Stockholm, Sweden, **(2009)** November 23-27, pp. 345-357.
[2] G. Friedrich, M. Fugini, E. Mussi, B. Pernici and G. Tagni, "Exception Handling for Repair in Service-based Processes", IEEE Transactions on Software Engineering, vol. 36, no. 2, **(2010)**, pp. 198-215.
[3] H. Abdeldjeli, N. Faci, Z. Maamar and D. Benslimane, "A Diversity-based Approach for Managing Faults in Web Services", Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, **(2012)** March 26-29, pp. 81-88.
[4] J.-P. Kim and J.-E. Hong, "Dynamic Service Replacement to Improve Composite Service Reliability", Proceedings of the 5th International Conference on Secure Software Integration and Reliability Improvement, Jeju Island, Korea, **(2011)** June 27-29, pp. 182-188.
[5] M. W. Zhang, Z. L. Zhu, D. C. Li, B. Zhang and L. Zhang, "An Execution Context Aware Approach for Web Service Substitution", Proceedings of the 7th International Conference on Information Processing and Management, Jeju Island, Korea, **(2011)** November 29-30,pp. 13-18.

[6] T. Espinha, C. Chen, A. Zaidman and H. G. Gross, "Maintenance research in SOA - Towards a standard case study", Proceedings of the 16th European Conference on Software Maintenance and Reengineering, Szeged, Hungary, **(2012)** March 27-30, pp. 391-396.
[7] H. Psaier and S. Dustdar, "A Survey on Self-healing Systems: Approaches and Systems", Computing, vol. 91, no. 1, **(2011)**, pp. 43-73.
[8] C. Zhang, H. P. Chen and J. B. Du, "A Tabu Search Approach for Dynamic Service Substitution in SOA Applications", Proceedings of the IEEE Asia-Pacific Services Computing Conference, Jeju Island, Korea, **(2011)** December 12-15, pp. 284-289.
[9] H. M. Ren and J. Liu, "Service Substitutability Analysis Based on Behavior Automata", Innovations in Systems and Software Engineering, vol. 8, no. 4, **(2012)**, pp. 301-308.
[10] H. Saboohi, A. Amini and H. Abolhassani, "Failure Recovery of Composite Semantic Web Services Using Subgraph Replacement", Proceedings of the International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, **(2008)** May 13-15, pp. 489-493.
[11] J. Kramer and J. Magee, "Analyzing Dynamic Change in Distributed Software Architectures", Proceedings of the 4th Biannual International Conference on Configurable Distributed Systems, Annapolis, MD, USA, **(1998)** October, pp.146-154.
[12] G. Santhanam, S. Basu and V. Honavar, "Web Service Substitution Based on Preferences Over Non-functional Attributes", Proceedings of the IEEE International Conference on Services Computing, Bangalore, India, **(2009)** September 21-25, pp. 210-217.
[13] R. Halima, K. Drira and M. Jmaiel, "A QoS-Oriented Reconfigurable Middleware for Self-Healing Web Services", Proceedings of the IEEE International Conference on Web Services, Beijing, China, **(2008)** September 23-26, pp. 104-111.
[14] Y. Zhai, J. Zhang and K. Lin, "SOA Middleware Support for Service Process Reconfiguration with End-to-End QoS Constraints", Proceedings of the IEEE International Conference on Web Services, Los Angeles, CA, United states, **(2009)** July 6-10, pp. 815-822.
[15] B. Benatallah, F. Casati and F. Toumani, "Representing, Analysing and Managing Web Service Protocols", Data and Knowledge Engineering, vol. 58, no. 3, **(2006)**, pp. 327-357.
[16] Y. Taher, D. Benslimane, M. Fauvet and Z. Maamar, "Towards an Approach for Web Services Substitution", Proceedings of the 10th IEEE International Database Engineering and Applications Symposium, Delhi, India, **(2006)** December 11-14, pp. 166-173.
[17] J. Pathak, S. Basu and V. Honavar, "On Context-specific Substitutability of Web Services", Proceedings of the IEEE International Conference on Web Services, Salt Lake City, UT, United states, **(2007)** July 9-13, pp. 192-199.
[18] Z. Song, X. Zhang and Y. Yin, "QoS Driven Service Dynamic Substitution Method", Computer Applications and Software, vol. 1, no. 9, **(2012)**, pp.27-30.
[19] D. Li, D. Cheung, X. Shi and V. Ng, "Uncertainty Reasoning Based on Cloud Models in Controllers", Computers and Mathematics with Applications, vol. 35, no. 3, **(1998)**, pp. 99-123.

Authors

Yan Gong is a senior engineer in Chinese Electronic Equipment System Engineering Corporation. He received his Ph.D. degree in computer science at Beijing University of Posts and Telecommunications of China in 2010. His research interests include service computing and distributed system.

Lin Huang is a Ph.D. student in Beijing University of Posts and Telecommunication. Her current research interests include service computing and cloud computing.

Fan Jiang is a senior engineer in Chinese Electronic Equipment System Engineering Corporation. He received his Master's degree in communication and electronic system at National University of Defense Technology in 1999. His research interests include communication software and distributed system.

Ke Han is a senior engineer in Chinese Electronic Equipment System Engineering Corporation. He received his Ph.D. degree in communication and electronic system at PLA University of Science and Technology of China in 1996. His research interests include network intelligence, communication software and distributed system.

