Applied Mathematics & Information Sciences
An International Journal

A Flexible Approach for Function-Consistent Service-Oriented Systems

Yuyu Yin[1], *Li Zhou*[1], *Xingjian Lu*[2] *and Ying Li*[2]

[1]College of Computer, Hangzhou Dianzi University, P.R.China
[2]College of Computer Science and Technology, Zhejiang University, P.R.China

Received: 28 Jul. 2012, Revised: 14 Oct. 2012, Accepted: 20 Oct. 2012
Published online: 1 Feb. 2013

Abstract: Service-oriented systems can be composed by different services from different providers. If one or more constitutes of services-oriented systems become fault at runtime, service-oriented systems cannot continue to serve their business function effectively. We propose an effective recovery approach for services based systems. The approach tries to replace each faulty service firstly. If some faulty services have not the candidate services with similar function, we will construct for each faulty service and tries to replace the region. Algorithm DRRank is designed to search for a substitution region that have a small number of replaceable services, some faulty and some healthy. Using our proposed approach, services-oriented systems can be recovered by replacing services in these selected regions rather than the whole service-oriented systems. As a result, we can reduce the computational complexity and the recovery overheads. At last, case study is given to show that how our approach works.

Keywords: Service-oriented Systems, Reconfiguration Region

1. Introduction

Service-Oriented Architecture(SOA) offers a powerful approach to build complex distributed systems by assembling many independently developed Web services in many applications[1]. As it is now adopted widely, great progress has been made in the research about service composition. Presently, service composition has become an increasingly important way for IT enterprises to rapidly develop their applications that not only satisfy customer business requirements but also deliver an expectable QoS[2,3].

With an increasing number of Web services, developing various service based systems(for short, SBSs) is not the only critical step of SOA implementation. Due to highly dynamic environments (e.g. in a clouding environment) and fast changing business requirements, one or more constituents of the SBS may become fault after a SBS assembled from a repository of component services has been deployed. In the event of faulty services, it is not desirable always stop and recompose the entire SBSs. Moreover, most enterprises would like to recover their applications with lower cost and better efficiency, so that their customers may not undergo as few unexpected business shutdowns as possible. Hence there arises an urgent need to provide function consistent SBSs to [4–10].

Up to now, although there existed many valuable works which focus on maintaining the overall functionality of SBSs in the presence of runtime environment changes[6]. Considering the recovery overheads, some researchers are considered that service substitution is the promising way[13,17]. However, it needs a necessary premise as follows: faulty services should have the candidate substitutions that offer the similar functionality. Once the premise is not satisfied, it is an additional problem of recovery of SBSs with lower overheads.

In order to address the problem, we present a dynamic recovery approach to handle multiple services failures and provide function consistent service-oriented systems. Inspired by our previous work[12], we introduce the notion of the substitution region to reduce the recovery overheads. The substitution regions are built for every faulty service. It includes faulty services and their neighbors. If two substitution regions overlap, they are combined into one. Then only the part of a SBS in the

* Corresponding author e-mail: yinyuyu@hdu.edu.cn

substitution region is replaced. In this way, the recovery process only involves a few component services.

The rest of this paper is organized as follows. Section 2 introduces our proposed recovery method and gives its algorithm. Section 3 gives the definition and the identification algorithm of the dynamic reconfiguration region. Section 4 gives the case study of our method and shows that how to get the dynamic reconfiguration region using the DRRank method. Section 5 surveys the related works. Section 6 gives the conclusions and future work.

2. Outline of Recovery Method for SBSs

The faulty services occurring in SBSs can trigger a recovery process. Firstly, we want to find the substitution services for all faulty services. If some faulty services have not any substitution, we construct and identify regions for each of them, and recompose the regions. Then, any region contains too many services, the whole service process will be recomposed. Figure 1 gives the process of our method.

For the clarity of the research, we found the candidates service with similar function by substitutability of services which studied in our previous work[12]. And Region recomposition use our proposed method reported in [23].

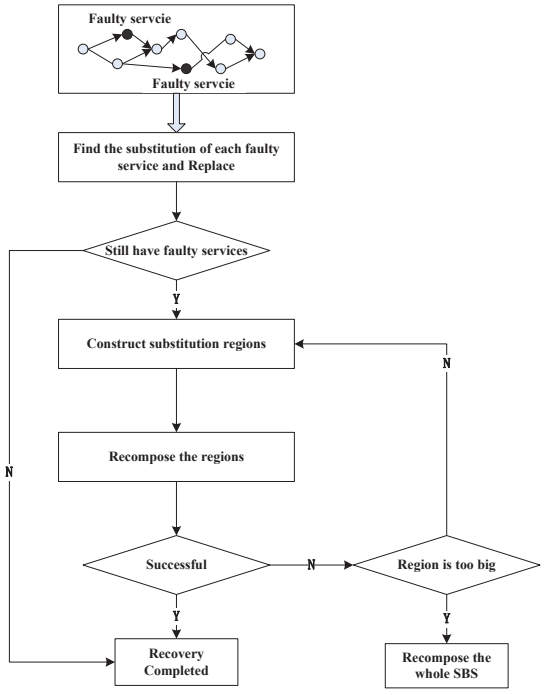

Figure 1 The process of our recovery method

3. Recovery Algorithm

The algorithm *RecSBS* gives the process of the recovery for SBSs. The algorithm would be triggered when the faulty services occurred in the systems. It firstly try to find the replacement for every faulty service(Step 2). If the replacement for some faulty services cannot be found, then we will construct substitution regions for these faulty services (step 6 to 10). The identification of substitution region will be discussed in session 3. After that, it will determine the size of the regions. If the size of the region is smaller than the threshold $\frac{N}{\alpha}$ (n is the number of components in a SBS, c is an overhead factor), the region will be recomposed using our previous method[23](step 17), else the whole SBS will be recomposed (step 11 to 15). If some regions still cannot find the compositions, the algorithm will increase the size of the regions, which means more nearby services will be added into the region.

Algorithm 1 RecSBS

Input: faulty services $S_f = \{s_i\}$
Output: Failure or Success

1: **for** each $s_i \in S_f$ **do**
2: **if** Find the replacement for each S_i **then**
3: $S_f = S_f - s_i$;
4: **end if**
5: **end for**
6: **SET** $c = 0$; // c is the region bound
7: **if** $S_f \neq \varnothing$ **then**
8: $c++$;
9: $fr = RegIden(S_f, c)$; //Call the Algorithm RegIden in section 3 to identify the substitution region
10: **while** $fr = \{r_i\} \neq \varnothing$ **do**
11: **if** the size of $r_i > \frac{N}{\alpha}$ **then**
12: Recompose the whole SBS;
13: **if** no composition can be found **then**
14: **return** Failure;
15: **end if**
16: **else**
17: Recompose r_i;
18: $fr = fr - r_i$;
19: **if** composition can be found **then**
20: $S_f = S_f - s'_i$; // s'_i is the faulty service in r_i;
21: **end if**
22: **end if**
23: **end while**
24: **if** $S_f \neq \varnothing$ **then**
25: **GOTO** 8;
26: **else**
27: **return** Success;
28: **end if**
29: **else**
30: **return** Success;
31: **end if**

The complexity of the algorithm depends on the while ”loop” and the identification algorithm of the substitution region (discussed in session 3).

4. Identify Substitution Region

In this section, the region identification algorithm is presented to identify the part of a SBS that should be replaced to recover SBS. By identifying a small substitution region, the number of the replaceable services will be reduced.

4.1. Definition of Substitution Region

A SBS may contain some different flow structure. The following four types of flow structures are considered in our study[12,14]: Sequence structure, Parallel structure, Selection structure, and Loop structure, as Figure 2. A substitution region is built for a faulty service and cannot cross the flow structures. Here, we give the definition of substitution region.

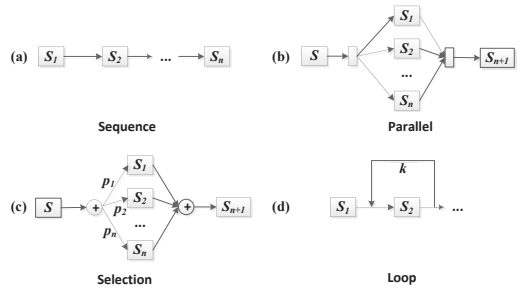

Figure 2 Structure Pattern

Definition 1. A substitution region r_i of service s_i is defined as a six tuples $r_i = SN, NB, SZ, Max\_Reg, F_b, F_e$, where:

(1). SN is the faulty service in r_i;

(2). NB a set of the services in r_i;

(3). SZ represents the size of r_i and $SZ \leq Max$.

(4). Max_Reg represents the max size of the region of the faulty service.

(5). F_b represents the start node of the flow structure which involves the faulty service.

(6). F_e represents the end node of the flow structure which involves the faulty service.

Definition 2. Maximum size Max of a substitution regions r_i of the faulty service s_i is defined as the furthest distance between the faulty service s_i and other service in the flow structure p_i. Namely,

$$Max\_Reg = Max(Distan_{P_i}(s_i, s_j)),$$

where, besides s_i , s_j is one of the rest of services in the flow structure p_i.

Definition 3. All substitution regions of the faulty service s_i are denoted as $SR_i = r_i, ..., r_n$.

4.2. Region Identification Algorithm

In this section, we give the identification algorithm of substitution region. The algorithm has two key processes as follows:

-Constructing all substitution region of a faulty service with the given bound;

-Identifying the closest region using the method in Section 3.3.

Algorithm *SReg_Ident* shows that how to construct and identify the substitution region for one service. The input for the algorithm includes the needed size of region d and the set S_f of all faulty services. For a faulty service s_i, its all regions can be found by Step 1 to 5. A region r_j can be built from the faulty service s_i to some other services that are connected to s_i with the size c of regions. In the region r_j, all services are not an end node of a flow structure. While such a region do not found, all services in the flow structure with the faulty services s_i should be added to the region r_j(Step 3 to 5). Next, the substitution factor of all regions of si will be computed by Formula (2), and then, a sort set of all regions of s_i is built by sorting the regions in descending order of substitution factor. Finally, if there are the same services in two regions, the algorithm merges the two into one (Step 13 to 17).

Algorithm 2 SReg_Ident

Input: faulty services $S_f = s_i$, the region size c.
Output: New Regions R for faulty services with bound c.

1: **for** each $s_i \in S_f$ **do**
2: **FIND** $r_j = s_j | \forall s_j \in r_j$, s_j is not a end node of a flow structure and the size of $r_j = c$;
3: **if** such r_j is not found **then**
4: **ADD** all nodes in the flow structure with faulty service s_i **TO** r_j;
5: **end if**
6: **ADD** r_j **TO** R_j
7: **end for**
8: **for** each $r_j \in R_j$ **do**
9: Calculate the substitution value of r_j using Formula (2);
10: **end for**
11: $R_j{}' =$**Sort all region** $r_j \in R_j$ in descending order of substitution value;
12: **ADD** R_j **TO** R
13: **for** $\forall r_j \in R_j$ **do**
14: **if** $r_j \bigcap r_k \neq \varnothing$ **then**
15: $r_j = r_j \bigcap r_k$
16: **end if**
17: **end for**
18: **return** R;

4.3. Substitution Factor

Due to no replacement for some faulty services, we need to construct substitution regions for each of them. For a faulty service, its regions may not only one. For example, In Figure 5, S_4 is a faulty service, and the size of the region is 3. As Definition 1, it can get three regions. As a result, we would like to search for the closest region. In the region, the invocations between all services occur more frequently.

In this paper, we propose the notion of substitution factor to identify the closest region. In order to address the problem, we propose the method DRRank which is based on PageRank method[16] to compute substitution factor.

PageRank computes the ranking of searchable pages based on a graph of the Web and it is an important method of the Google search. Google uses it to reflect the relevance and importance of the web pages. PageRank is defined as follows: PageRank assumes that a Web user will eventually stop clicking any link. The probability[15] is known as a damping factor $q \in [0,1]$, which is set to about 0.85. Suppose a webpage u is pointed by a set of pages $M(u)$, and u points to a set of pages $L(u)$. The page rank $PR(u)$ of the page u is given by the following formula:

$$PR(u) = \frac{1-q}{N} + q \times \sum_{v \in M(u)} \frac{PR(v)}{L(v)} \quad (1)$$

Figure 3 shows a graph of three webpages A, B, and C which link each other. Based on the Formula (1), we can get the PageRank result of the three pages $PR(A) \approx 1.27$, $PR(B) \approx 1.67$, $R(C) \approx 1.5$.

$$PR(A) = \frac{1-0.85}{3} + 0.85 \times (\frac{PR(C)}{1})$$
$$PR(B) = \frac{1-0.85}{3} + 0.85 \times (\frac{PR(A)}{1} + \frac{PR(C)}{2})$$
$$PR(C) = \frac{1-0.85}{3} + 0.85 \times (\frac{PR(B)}{1})$$

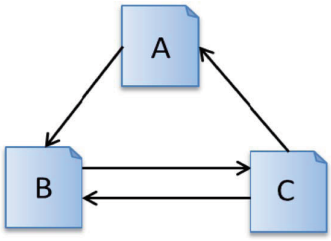

Figure 3 Link Analysis

The connections between the component services in a SBS can be viewed as the links between the web pages. Because of the similar features between the web services and the web pages, PageRank has been used in web services search field[16]. The substitution regions are composed by some interdependent services. As a result, we can use the idea of the PageRank to evaluate the close level between the services in the same region.

The Formula of our proposed method DRRank is as follows:

$$DR(WS) = \frac{1-q}{N} + q \times \sum_{v \in M(ws)} \frac{DR(v)}{L(v)} + \frac{P_{out}}{P_{all}} \quad (2)$$

We use ws to represent the region. In Formula (2), $M(ws)$ means a set of pages point to ws, and ws point to a set of pages $L(ws)$. Because of many series in a SBS, the importance of the component services also relates to the size of the input and output parameters. P_{out} denotes the size of the external links of ws. P_{all} denotes all the links of ws, including the internal links between the component services in ws and the external links with other services. The proportion of the P_{out} and P_{all} expresses the complexity of ws. The bigger the values is, the more complexity the ws is. The value of substitution factor means the recovery complexity of the regions. The bigger the DR value is, the higher the recovery costs are. Therefore, we should choose the region with the lower DR value to be recomposed.

From Formula (2), we can see that the equation of $DR(ws)$ is similar to the PageRank method. Thus, They have similar convergence and complexity. And the proportion of the P_{out} and P_{all} can be given.

5. Case Study

In this section, an example is given to explain how the region identification algorithm works.

In the region identification process, the component services in the same flow structure will be combined firstly to be a region. As Figure 4, S_2 is a faulty service. When c is set to 2, it can find two regions S_2, S_3 and S_1, S_2. Because S_1 and S_2 are not in the same flow structure, the algorithm Reg_Iden can only identify the region S_2, S_3.

In Figure 5, S_4 is a faulty service. When c is set to 3, we can find three regions S_2, S_3, S_4,S_3, S_4, S_5 and S_4, S_5, S_6. And we should use the DRRank method to evaluate their substitution factor.

Value of the revlant parameters of the component services in Figure 4 are as Table 1. In the table, DR means the initial DR value of every component services. I means the size of the input parameters of every component services. O means the size of the output parameters.

According to Formula (2), we can get the equation of substitution factor of the three regions as follows:

$$DR(1) = \frac{1-q}{N} + q \times (\frac{DR(S_1)}{2}) + \frac{I_{S_2} + O_{S_4}}{IO_{S_2} + IO_{S_3} + IO_{S_4}} = \frac{1-0.85}{7} + 0.85 \times (\frac{0.83}{2}) + \frac{6}{20}$$
$$DR(2) = \frac{1-q}{N} + q \times (\frac{DR(S_2)}{1}) + \frac{I_{S_3} + O_{S_5}}{IO_{S_3} + IO_{S_4} + IO_{S_5}} = \frac{1-0.85}{7} + 0.85 \times (\frac{0.15}{1}) + \frac{6}{16}$$

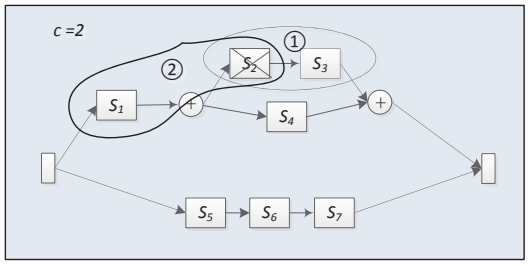

Figure 4 Combination Priority

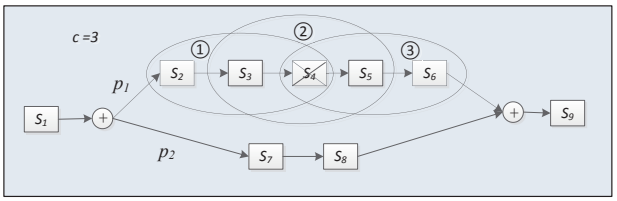

Figure 5 Identification of the region with $c = 3$

Table 1 The Parameters of The Component Services

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
DR	0.83	0.15	0.47	0.62	0.38	0.67	0.27	0.55	0.72
I	2	4	4	3	2	2	4	4	5
O	2	4	3	2	2	5	4	5	3

$$DR(3) = \frac{1-q}{N} + q \times \left(\frac{DR(S_3)}{1}\right) + \frac{I_{S_4} + O_{S_6}}{IO_{S_4} + IO_{S_5} + IO_{S_6}} = \frac{1-0.85}{7} + 0.85 \times \left(\frac{0.47}{1}\right) + \frac{8}{19}$$

The DR values of the three regions are as Table 2. Reg.a represents the region S_2, S_3, S_4. Reg.b represents the region S_3, S_4, S_5. Reg.c represents the region S_4, S_5, S_6. The region with the lowest DR value is S_2, S_3, S_4.

Table 2 The *DR* Value of The Region

region	Reg.a	Reg.b	Reg.c
value	0.318	0.383	0.441

6. Related Work

Service-oriented systems are often executed in dynamic environments. In the environment, many factors can interrupt the execution of the applications, such as unavailability or malfunctioning of services. Therefore, the recovery of the service-oriented systems is very important.

For the faulty services in SBSs, many papers have proposed service substitutability and adaptation strategy. T. Yu[17] proposes two service substitutability algorithms. The first one uses an existing backup path. Once the service is failed, the system can turn to the backup path quickly. The second one is to recompose a new path and replace the old one. Khaled Mahbub[18] analysis the different situation about the failure of the service-oriented systems, and proposes a service substitutability strategy to support the dynamic changes of the systems. The strategy can determine the timing of the executing, which can be before or after the execution. Wei Ren[19] proposes adaptive strategy to support the dynamic service substitutability which is using business rules. It uses local and global business rules to choose the candidate services to replace ones. The most of the work just consider one-to-one substitutability. With the development of the coarse-grained services, the service substitutability can be one-to-one, one-to-many and many-to-one.

Hadi Saboohi etc.[20] propose a failure recovery method using subgraph replacement of web services containing a failed web service. They first calculate the subgraph of the composition service, and then rank the subgraphy. The best ranked alternative subgraph would be the replacement selection if the web service of that group fails. The subgraph calculation considers all combination patterns of all the component services in the composition service, so it is time-consuming. Yanlong Zhai etc.[11] present an approach for repairing failed services and still meets the user's specified end-to-end QoS constraints. The approach is designed to produce reconfiguration regions that include one or more failed services. Then the approach uses integer programming to recompose each region until the reconfiguration is success. Kwei-Jay Lin[21,22] recompose sub-process by considering more function replacement models. Services in reconfiguration regions may be replaced using one-to-one, one-to-many, or many-to-one service mappings. The paper proposes the algorithm to recompose sub-process with many-to-one mappings. In our paper, we consider dynamic reconfiguration regions. Based on the regions, the dynamic reconfiguration method uses service substitutability strategy or recomposition strategy to reconfigure the system according to the size of the region.

7. Conclusion and Future Work

Since service-oriented systems are executing in highly dynamic environments, they need to recover from service faults as soon and efficiently as possible. As a result, maintaining the high availability of service-oriented system is listed as the obstacle for the next development phase of SOA and cloud computing and it has become the first challenge in the tow areas.

When the faulty services appear in the system, it is undesirable to recompose the whole system if there are

only a few faulty services. We propose an effective recovery approach for services based systems. The approach tries to replace each faulty service firstly. If some faulty services have not the candidate services with similar function, we will construct for each faulty service and tries to replace the region. Algorithm DRRank is designed to search for a substitution region that have a small number of replaceable services, some faulty and some healthy. Using our proposed approach, services-oriented systems can be recovered by replacing services in these selected regions rather than the whole service-oriented systems. As a result, we can reduce the computational complexity and the recovery overheads.

In the future, our work could also be extended in the way: the identification process of the substitution region should be improved. In addition, our approach should be applied to more real applications.

Acknowledgement

This paper is granted by National Natural Science Foundation of China under Grant No. 61100043, Zhejiang Provincial Natural Science Foundation(No. LY12F02003), and The National Key Technology R&D Program under Grant (No. 2012BAH24B04).

References

[1] LiangJie Zhang, Jia Zhang, and Hong Cai, Services Computing, Springer&Tsinghua University Press, 2007.

[2] Zhang DG and Zhang XD. A New Service-Aware Computing Approach for Mobile Application with Uncertainty. APPLIED MATHEMATICS & INFORMATION SCIENCES, 2012, 6(1):9-21.

[3] M. Brian Blake, Wei Tan, Florian Rosenberg, Composition as a Service. IEEE Internet Computing (INTERNET), 2010, **14(1)**: 78-82.

[4] M. Aoyama, S. Weerawarana, H. Maruyama, C. Szyperski, K. Sullivan, and D. Lea, Web Services Engineering: Promises and Challenges, Proc ICSE 2002, Orlando, 2002, 647-648.

[5] W. Vambenepe,C. Thompson,V.Talwar et.al, Dealing with Scale and Adaptation of Global Web Services Management. Int. J. Web Service Res, 2007, **(3)**: 65-84.

[6] Yuhong Yan, Pascal Poizat, Ludeng Zhao, Repair vs. Recomposition for Broken Service Compositions. ICSOC 2010: 152-166.

[7] Bo Jiang, W. K. Chan, Zhenyu Zhang, T. H. Tse,Where to adapt dynamic service compositions. WWW 2009: 1123-1124.

[8] Liu, Xumin, et al. "Ev-LCS: A System for the Evolution of Long-Term Composed Services." IEEE Transactions on Services Computing (2011).

[9] Friedrich, Gerhard, et al. "Exception handling for repair in service-based processes." IEEE Transactions on Software Engineering, 2010, **(36)2**: 198-215.

[10] Romano D, Pinzger M. "Analyzing the Evolution of Web Services Using Fine-Grained Changes 2012 IEEE 19th International" Conference on Web Services (ICWS), 2012: 392-399.

[11] Yanlong Zhai, Jing Zhang, Kwei-Jay Lin.SOA Middleware Support for Service Process Reconfiguration with End-to-End QoS Constraints. 2009 IEEE International Conference on Web Services(ICWS), 2009: 815-822.

[12] Ying Li, Xiaorong Zhang, Yuyu Yin ,Yuanlei Lu. Towards Functional Dynamic Reconfiguration for Service-Based Applications. The IEEE World Congress on Services(Services 2011): 467-473.

[13] Wu J and Jin L. A Linear Logic Representation for BPEL Process Protocol. APPLIED MATHEMATICS & INFORMATION SCIENCES, 2011, **5(2)**: 25-31.

[14] Ying Li, Xiaorong Zhang, Yuyu Yin, Jian Wu. QoS-Driven Dynamic Reconfiguration of the SOA based Software. International Conference on Service Sciences (ICSS). 2010.

[15] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 1998, **30** (1-7): 107-117.

[16] Li jun Mei, W.K. Chan, T.H. Tse.An Adaptive Service Selection Approach to Service Composition. 2008 IEEE International Conference on Web Services.

[17] T. Yu and K.Lin. Adaptive algorithms for Finding Replacement Services in Autonomic Distributed Business Processes.In Proceedings of the 7th International Symposium on Autonomous Decentralized Systems, 2005: 427-434.

[18] Khaled Mahbub, Andrea Zisman. Replacement Policies for Service-BasedSystems. Service-Oriented Computing, ICSOC/ServiceWave Workshops 2009: 345-357.

[19] Wei Ren, Gang Chen, Zhonghua Yang, etc. Self-healing Capable Workflow Execution with Semantic Web Service. IEEE International Conference on Service Operations and Logistics and Informatics (IEEE/SOLI 2008): 508-513.

[20] Hadi Saboohi, Amineh Amini, Hassan Abolhassani. Failure Recovery of Composite Semantic Web Services using Subgraph Replacement. Proceedings of the International Conference on Computer and Communication Engineering (ICCCE 2008): 489-493.

[21] Kwei-Jay Lin, Jing Zhang, Yanlong Zhai. An Efficient Approach for Service Process Reconfiguration in SOA with End-to-End QoS Constraints. IEEE Conference on Commerce and Enterprise Computing (CEC 2009): 146-153.

[22] Kwei-Jay Lin, Jing Zhang ,Yanlong Zhai ,Bin Xu. The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA. Service Oriented Computing and Applications (SOCA 2010) **4(3)**: 157-168.

[23] Bin Wu, ShuiGuang Deng, Ying Li, Jian Wu, Jianwei Yin, "AWSP: An Automatic Web Service Planner Based on Heuristic State Space Search". ICWS 2011, (IEEE Press, 2011): 403-410.

Yuyu Yin received the Doctor's degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He is currently an assistant professor in Hang-zhou Dianzi University. His research interests include service computing, cloud computing and middleware techniques.

Ying Li received the Doctor's degree in software technology from Zhejiang University, Hangzhou, China, in 2000. He is currently an associate professor in Zhejiang University. His research interests include software architecture, software automation, compiling technology, and middleware techniques.