International Journal of Grid Distribution Computing
Vol.8, No.2 (2015), pp.133-142
http://dx.doi.org/10.14257/ijgdc.2015.8.2.13

Research on Semantic Web Service Composition Based on Binary Tree

Shengli Mao, Hui Zang and Bo Ni

Computer School, Hubei Polytechnic University, Huangshi, China
maoshengli123@163.com

Abstract

With the rapid development of cloud computing and service computing, Web services are combined to form the composite service with a large granularity is an important research direction at present. The current service composition methods have the problems of low composition efficiency and accuracy, this paper proposes a semantic Web service composition method based on the binary tree. This method uses the ontology semantic reasoning relationship and binary tree theory to composite Web services. The composition relations between Web service interfaces are mainly considered. The approach can enhance the efficiency and accuracy of service composition, and the experiments are used to validate and analyze the proposed methods.

Keywords: *Web Service, Composition, Binary Tree, granularity*

1. Introduction

In the modern era of service-oriented software engineering, the technologies about Web services get more and more attention in the researcher work [1, 2]. Service composition refers to assemble atomic services to form composite services with complicate functions and large granularity according to service relations and user's requests. This can help to meet user's requirements better [3].

There is some research work about service composition at present, and the research work uses all kinds of methods and different mechanisms to realize service composition. In mainly includes the following aspects: realizing service composition according to user's specific requests and the relationship between them. For example, some methods use the formal methods to verify the service composition and validation; some methods realize service composition from the non-functional level. These methods realize service composition from different aspects and they can enhance service composition efficiency. But these methods don't consider the semantic relationship and the composition accuracy will be influenced. At the same time, they don't consider the semantics when do the matching calculation according to user's requests. This leads to users can't find the needed composite services accurately. In addition, the efficiency of these service composition methods is very low.

In order to solve above problems, this paper proposes a semantic Web service composition method based on binary tree. This method uses ontology to annotate service interface firstly, then it does the matching calculation from the semantic level to find the services that can be composited. The binary tree is used to construct the composite services with large granularity, and it can help enhance efficiency and accuracy of service composition. The operations (like find, merge nodes) about binary tree can be used to find the atomic and a set of composite services with correlation quickly to meet user's personal requirements. This method can enhance service discovery efficiency.

ISSN: 2005-4262 IJGDC

In Section 2, we introduce the overall architecture of our proposed methods. How to use binary tree theory to realize semantic Web service composition is given in section 3. In Section 4, we discuss the related research work of service composition. We use experiment to analyze and validate the proposed method in Section 5. Finally, the conclusion and the future research work are discussed.

2. Overall Architecture

This paper presents a Web service composition approach on the basis of binary tree, and its architecture is shown in Figure 1. It mainly includes the following aspects: ontology annotation, service composition, binary tree merge, request analysis, service selection.

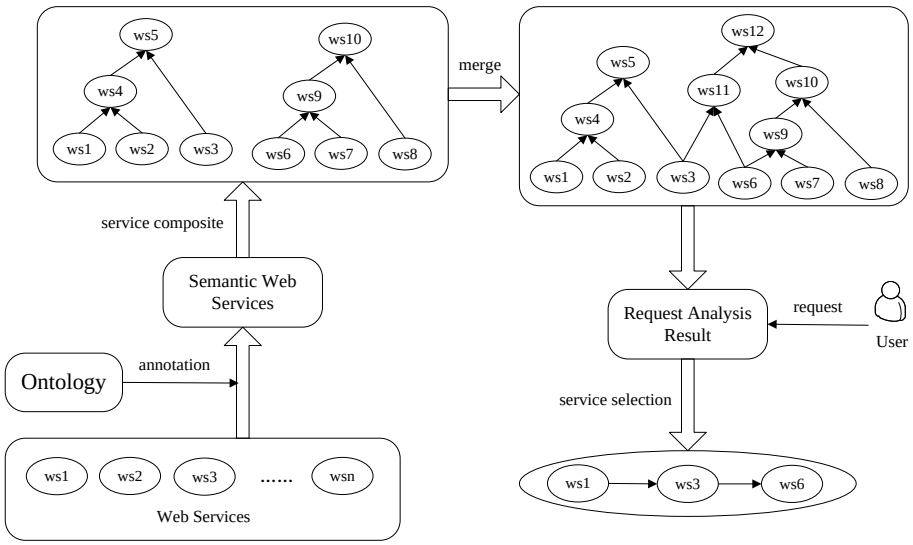

Figure 1. Semantic Web service Composition Based on Binary Tree

In the above figure, it uses concepts and semantic relations among concepts in ontology to annotate the interface of Web services firstly, and this can lay the foundation of matching calculation from the semantic level. Then it does matching calculation between service input and output to determine the composition relationship between services. Then it uses the basic binary tree theory to construct composite service tree. The services which realize specific topic can form service binary tree with small size, and these trees can be merged to form binary tree with large size. The corresponding services can be composited to realize more functions and topics. Finally, the user's requests will be analyzed, and this method can discover the atomic service and composite services with the correlations in the composite service binary tree. The efficiency and accuracy of service composition and service discovery can be improved.

3. Semantic Web Service Composition based on Binary Tree

3.1. Binary tree and ontology semantic

Binary Tree

Tree is a kind of important nonlinear data structure, and it is a data element (in the tree are called nodes) according to the structure of the branch relations organization. Binary tree is an ordered tree which has at most two subtrees of each node. The root of the subtree is called left subtree and right subtree. Each node of the binary tree has only two subtrees at most, and it means there is no node whose degree is more than 2. The subtree of binary tree has the division of left and right, and the order can't be reversed.

Ontology Semantic Reasoning

Ontology can be used to describe concepts and the semantic relations between concepts. It includes concepts, properties, instances, concept relations, rules *etc.*, and it can supply the formal definition and axiom.

Definition 1. Ontology=$\{C_{set}, R_{set}\}$

- $C_{set}=\{c_i, i\in 1, 2...n\}$, c_i is a concept in C_{set} of Ontology.
- $R_{set}=\{<c_i, r_x, c_j>, c_i, c_i \in C_{set}, i, j \in 1, 2...n, r_x \in \{SubclassOf, SuperClassof, Intersectionof\}, x\in 1, 2...m\}$, R_{set} is the concept relation set of Ontology.

OWL is an ontology description language which is used mostly; we can use commonly ontology reasoners (such as OWL-API, Pellet, Hermit, Jena) to do semantic reasoning and get results based on the specific rules. The semantic relations between concepts mainly include Equivalent, SuperClassOf, SubClassOf, Intersection and Fail. They can represent the relations of Exact, Plugin, Subsume, Intersect and Fail [4] between concepts.

Web Service

In the consideration of input and output of Web service, we use ontology concepts to annotate the service interfaces. The definition is shown in the following Definition 2.

Definition 2. Web Service(*ws*)=$\{Input, Output\}$

- $Input=\{I_i, I_i\in C_{set}, i\in 1, 2...n_i\}$, I_i is an input element in *Input* of *ws*.
- $Output=\{O_j, O_j\in C_{set}, j\in 1, 2...n_j\}$, O_j is a output element in *Output* of *ws*.

3.2. Web Service Composition Algorithm

On the basis of the above definition, the following Algorithm 1 gives the process of how to realize semantic Web service composition using the binary theory.

Algorithm 1. Web service composition algorithm
Input: *Ontology*, $WS_{set}=\{ws_i, i=1,2,\ldots snum\}$
Output: $Btree_{ws}$
1: $Btree_{ws}\leftarrow\varnothing$, $val\leftarrow 0$, node$\leftarrow\varnothing$
2: foreach *web service* $ws_i\in WS_{set}$
3: Using Ontology in annotate the interface of ws_i in WS_{set}
4: *node*←new Node(ws_i.*Input*, ws_i.*Output*)
5: $Btree_{ws}$.add(*node*)
6: end foreach
7: foreach *web service* $ws_i\in WS_{set}$
8: foreach *web service* $ws_j\in WS_{set}$
9: if(ws_i != ws_j) then
10: *val*←*matchIO*(ws_i.*Output*, ws_j.*Input*)

11: if($val > \alpha$) then
12: *node*←new Node(ws_i.*Input*, ws_j.*Output*)
13: *node.leftchild*←ND(ws_i)
14: *node.rightchild*←ND(ws_j)
15: $Btree_{ws}$.add(*node*)
16: Get the realizing topic of service ws_i and ws_j to composite
17: end if
18: end foreach
19:end foreach
19:Similar to step 7~19, merge the new nodes in $Btree_{ws}$
20:return $Btree_{ws}$

The above algorithm gives the process of how to use the binary tree theory to realize semantic web services composition. The initialization work is done through step 1. This algorithm uses the ontology to annotate the service interfaces in WS_{set} firstly. It will create a tree node for each service and add it into $Btree_{ws}$, as seen in step 2-6. The interface (Input and Output) of every two services (Ser_A, Ser_B) are done matching calculation. The services whose matching value is more than the threshold will be constructed as a new tree node. Its left subtree is Ser_A and its right subtree is Ser_B. The input of new composite service node is the input of Ser_A and its output is the output of Ser_B. The corresponding topic will be determined, as seen in step 7-19. Using the same method, we can get more binary trees and merge these binary trees into a large binary tree which realizes more complicate and large topics. Finally, return $Btree_{ws}$.

3.3. Web Service Selection Algorithm

Based on service composition, how to realize service selection according to user's request will be given in the following Algorithm 2. It can find the atomic and a set of services with correlation between them to meet user's personal requirements.

Algorithm 2. Web service selection algorithm
Input: *RE*, $Btree_{ws}$, WS_{set}
Output: *rws*
1: $rws \leftarrow \varnothing$, $ws \leftarrow \varnothing$, *lenode*$\leftarrow\varnothing$, *rinode*$\leftarrow\varnothing$
2: foreach *web service* $ws_i \in WS_{set}$
3: if((*matchIO*(*RE.ReInput*, ws_i.*Input*)+*matchIO*(*RE*. *ReOutput*, ws_i.*Output*))/2 > α) then
4: $rws \leftarrow rws \cup ws_i$
5: end if
6: end foreach
7: Find the binary tree to realize the *RE*
8: foreach *node* $no_i \in Btree_{ws}$
9: *ws*←*Change*($Btree_{ws}$.no_i)
10: if((*matchIO*(*RE.ReInput*, *ws.Input*)+*matchIO*(*RE*. *ReOutput*, *ws.Output*))/2 > α) then
11: *lenode*←no_i.*leftnode*
12: *rinode*←no_i.*rightnode*
13: find the subnode iteratively until the bottom node
13: $rws \leftarrow rws \cup$ *Change*($Btree_{ws}$.*lenode*)
14: $rws \leftarrow rws \cup$ *Change*($Btree_{ws}$.*rinode*)
15: end if
16:end foreach
17:return *rws*

In the above algorithm, *RE* represents user's request. It selects the atomic service which can meet user's request firstly. The matching calculation is done from the aspect of Input and Output to select the services whose matching value is large than the threshold, as seen in step 2-6. When to find composite services, the binary tree which is interested for users will be selected firstly, and this can narrow the service search scope, as seen in step 7. On the basis of searching the nodes which can meet user's request, it finds the left and right subtree nodes of the node recursively till the nodes in the lowest layer. Then it adds services of these nodes into *rws*. Finally, return *rws*.

4. Related Work

There are lots of research work about web service composition and service discovery. The service discovery methods are often used on the basis of particular service composition approach and users can find the composited services to meet their needs efficiently according to specific requests. The service composition research work mainly concentrates on the following two aspects: the first one refers to composite Web services according to users' requirement; the second one refers to find the composite services quickly based on compositing different services. This paper mainly concentrates on the second research work.

Lee, *et al*., propose a scalable and efficient web services composition method based on a relational database [5], they develop a web services composition search system called PSR. The PSR system stores the graph into tables and computes answers for semantic web services composition search in advance by joining Tables. In [6], the authors present an approach realizing larger granularity service composition for business users, which is based on prefabricated and modifiable templates and constraint solving. The business users can construct applications just like assembling hardware by composing larger granularity and reusable modules. In [7], Wang, *et al*., have proposed a service composition method for tradeoff between satisfactions of multiple requirements. Tradeoff strategies are proposed for genetic algorithm and a service composition method is put forward for tradeoff between satisfactions of multiple service requirements. The authors in [8] introduce a user-oriented approach which aims to simplify service composition. The authors leverage the plentiful information residing in service tags, both from service descriptions (such as WSDL) and the annotations tagged by users.

Ding, *et al*., in [9] present a use-centric service composition method synthesizing multiple views. The user's requirements are transferred to operations on multiple views through business data. And the corresponding service composition construction algorithm has been proposed for immediate decision-making. In [10], Hwang, *et al*., propose a dynamic web service selection approach for reliable web service composition, and the finite state machine theory is used to realize service composition for users. Yang, *et al*., in [11] propose a QoS pruning-based top-k automatic service composition method. A forward service filtering algorithm is employed for reducing solution spaces and a greedy-based pruning algorithm is designed for backward searching for Top-k QoS optimal solutions efficiently. Zhang, *et al*., have proposed a web services outsourcing manager framework via a mathematical model for dynamic business processes configuration using existing web services to meet customers' requirements [12]. They use a novel mechanism to map a service selection problem into a solution space {0, 1} to utilize global optimization algorithms such as Genetic Algorithms (GA).

The above research work uses the different approaches and mechanisms to composite services, but some of these approaches don't consider the semantic features of services. This leads to low efficiency and accuracy of service composition. When to realize

service composition, the efficiency and accuracy of some approaches are too low to influence the performance. Based on semantic annotation of service interface using ontology, we use the binary tree theory to organize services which have the composition relations. The composite service with large granularity can be formed, and the efficiency and accuracy of service composition and discovery can be enhanced.

5. Experiment

5.1. Experiment Environment

We ran our experiments on an AMD A6-4400M APU with Radeon(tm) HD Graphics machine with 2 GB memory running Windows7 Professional. Our algorithms were implemented using Java and MySQL. We mainly use the publicly available service retrieval test collection OWLS-TC v4 (http://projects.semwebcentral.org/projec ts/owls-tc/). This dataset includes more than 1000 services in the format of .wsdl and .owls. And the corresponding user's request and ontology are also included, and the concepts in the all the ontologies can be used to annotate to services in the dataset.

5.2. Experiment Analysis

This paper proposes a kind of semantic Web service composition method based on binary tree theory. This section designs some experiments to compare the time and accuracy of service composition and discovery of the proposed method with other approaches.

Experiment 1. Semantic annotation time

This experiment mainly compares the time of using ontology to annotate different numbers of services semantically. In the condition of following different number of services: 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000, we annotate the interface (Input and Output) of services and statistic the using time. The experiment result is shown in Table 1.

Table 1. Comparison of Semantic Annotation Time of Different Service Numbers

service numbers	50	150	250	350	450	550	650	750	850	1000
Time(s)	13	30	51	74	89	108	134	174	194	213

Through Table I we can see the service interface annotation time is largely different in the case of different service numbers. The using time of becoming large as the number of services increases. In addition, we can see the interface annotation time of 10 services is about 2s.

Experiment 2. Comparison of service composition time and numbers

On the basis of service annotation, this experiment compares and analyzes the service composition time and number in the case of using binary tree theory to composite Web services with different numbers. The experiment is done in the following service numbers: 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. The experiment result is shown in Figure 2 and 3.

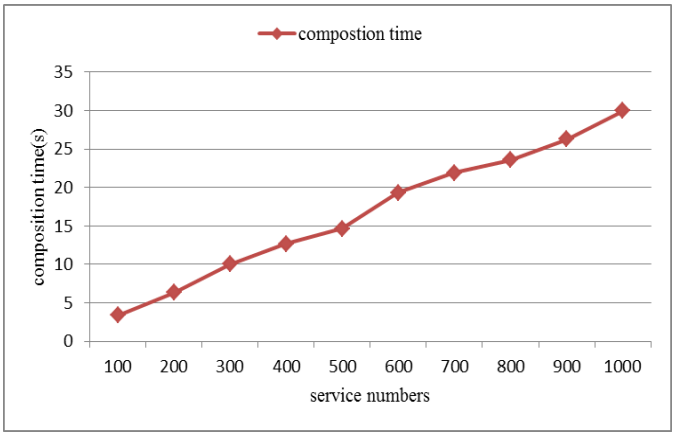

Figure 2. Comparison of Web Service Composition Time

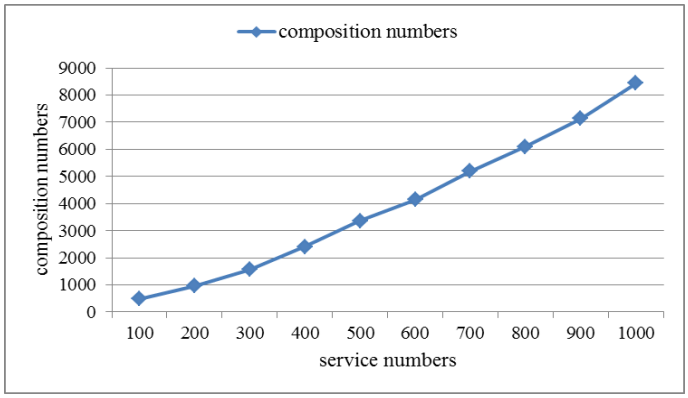

Figure 3. Comparison of Web Service Composition Numbers

Through the above Figure we can see the time of service composition is different in the case of different service numbers. The service composition time is gradually increased as the number of services increases. The time of compositing 100 services is about 3s. In addition, the number of composite service shows the trend of rapid growth, and most of the services can be combined.

Experiment 3. Comparison of service composition accuracy of different semantic relations

In the case of different service numbers: 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000, this experiment compares and analyses the service composition accuracy in the condition of different semantic relations: EquivalentOf, SubClassOf, SuperClassOf and Intersection. The experiment result is shown in Figure 4.

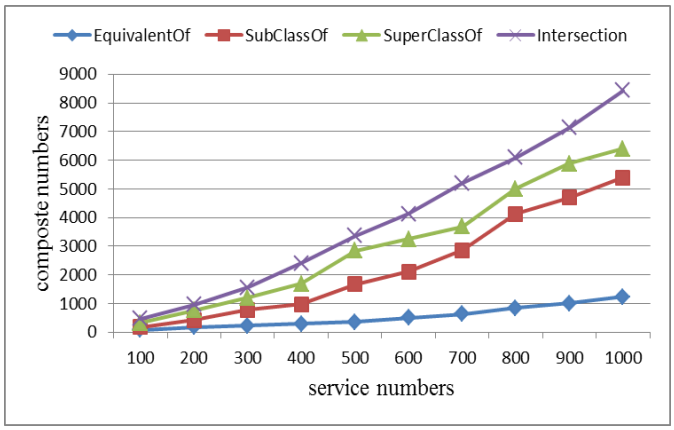

Figure 4. Comparison of Web Service Composition Numbers of Different Semantic Relations

Through Figure 4 we can see the number of composite service gets rapidly growth as the service number increases and in different semantic relations. The number of composite service is the most of all when the semantic relationships are Intersection between concepts. The number is the least of all when the relationship is EquivalentOf, and SubClassOf, SuperClassOf is followed. This is because it only considers the equivalent relationship and the number of matching service will be reduced. When the relationship is Intersection, it considers all the different semantic relations between concepts and the number of matching services will be increased.

Experiment 4. Comparison of service discovery time and accuracy

It helps to find the composite services accurately and quickly after realizing semantic Web services composition. Based on realizing service composition using binary tree theory, we call the method which finds the atomic and composite services according to user's request as the BTComposition method. And the Random method means it does not use any composition methods for service composition and discovery. When the number of services is 1000, we use the above two methods to deal with the services and find the needed services. In the case of different numbers of service requests: 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, we compare and analyze the time and accuracy of service discovery. The result experiment is shown in Figure 5 and 6.

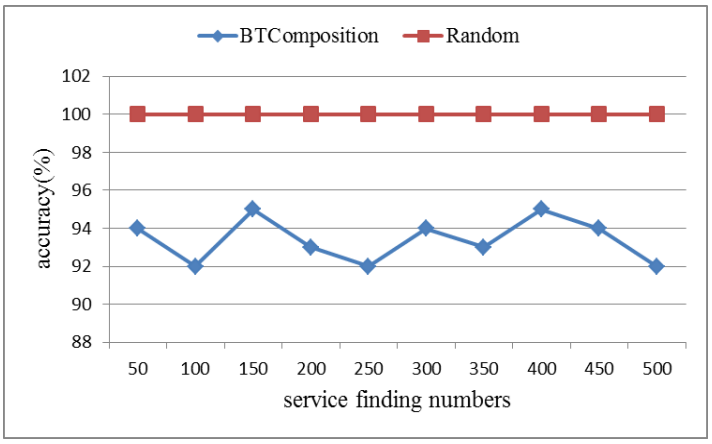

Figure 5. Comparison of Service Finding Accuracy

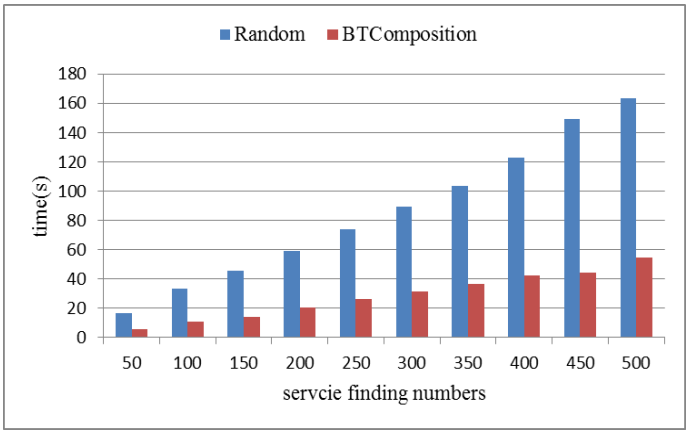

Figure 6. Comparison of Service Selection Time

Through the figure we can see the service finding discovery accuracy of BTComposition method is less than Random method, and it is about 92%~96%. This is related to the service composition accuracy of BTComposition method. In addition, the service discovery efficiency of BTComposition method is largely than Random method apparently. The using time of the former method is less than the latter; this is because the services are combined through the BTComposition method. And users can quickly locate the binary tree which realize the specific topic, thus the efficiency of finding atomic service and composite service can be enhanced.

6. Conclusion

In service computing, the atomic services can't meet user's personal requirements usually and it needs to find composite services according to specific request. In order to solve the problem of low efficiency and accuracy of service composition and service discovery, this paper uses the binary tree theory to composite semantic Web services on the basis of annotating Web services on the internet. The composite services with large granularity can be formed. It also discusses how to find the atomic services and set services which can be composited accurately and efficiently according to user's request. Finally, it uses experiments to validate the proposed methods.

The future research work mainly includes the following aspects: composite and discover the semantic Web services from the aspect of services process; Using document topic generation model-LDA method to composite services in further based on the binary tree service composition, and it helps to realize service composition and discovery efficiently; Consider the service clustering and non-functional properties of services to enhance service discovery efficiency in further.

Acknowledgements

This research project was supported by the Educational Commission of Hubei Province of China under grant No. b2014034.

References

[1] M. P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-Oriented Computing: A Research Roadmap", International Journal of Cooperative Information Systems, vol. 17, no. 2, **(2008)**, pp. 223-255.

[2] K. Gottschalk, S. Graham, H. Kreger and J. Snell, "Introduction to Web services Architecture", IBM Systems Journal, vol. 41, **(2002)**, pp. 170-177.
[3] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, "Unraveling the Web Services Web: an Introduction to SOAP, WSDL, and UDDI, Internet Computing", IEEE, vol. 6, **(2002)**, pp. 86-93.
[4] M. Barhamgi, D. Benslimane and B. Medjahed, "A Query Rewriting Approach for Web Service Composition", IEEE Transactions on Services Computing, vol. 3, no. 3, **(2010)**, pp.206-222.
[5] D. Lee, J. Kwon, S. J. Lee, S. Park and B. Hong, "Scalable and Efficient Web Services Composition Based on a Relational Database", Journal of Systems and Software, vol. 84, **(2011)**, pp. 2139-2155.
[6] H. T. Hu, G. Li, Y and B. Han, "An Approach to Business-User-Oriented Larger-Granularity Service Composition", Chinese Journal of Computers, vol. 28, no. 4, **(2005)**, pp. 694-703.
[7] X. Z. Wang, Z. J. Wang, X. F. Xu and Y. Liu, "A Service Composition Method for Tradeoff Between Satisfactions of Multiple Requirements", Journal of Computer Research and Development, vol. 48, no. 4, **(2011)**, pp. 627-637.
[8] X. Z. Liu, G. Huang and H. Mei, "A User-Oriented Approach to Automated Service Composition", IEEE International Conference on Web Services, **(2008)**, pp. 773-776.
[9] W. L. Ding, Q. Wang and S. Zhao, "A User-Centric Service Composition Method Synthesizing Multiple Views", Chinese Journal of Computers, vol. 34, no. 1, **(2011)**, pp. 131-142.
[10] S. Y. Hwang, E. P. Lim, C. H. Lee and C. H. Chen, "Dynamic Web Service Selection for Reliable Web Service Composition", IEEE Trans. on Services Computing, vol. 1, no. 2, **(2008)**, pp. 104-116.
[11] R. T. Yang, S. Q. Zhang and W. C. Dou, "A QoS Pruning-Based Top-k Automatic Service Composition Method", ACTA electronica sinica, vol. 40, no. 7, **(2012)**, pp. 1489-1491.
[12] L. J. Zhang and B. Li, "Requirements Driven Dynamic Services Composition for Web Services and Grid Solutions", Journal of Grid Computing, vol. 2, **(2004)**, pp. 121-140.

Authors

Mao Shengli, he is a lecture in Computer School of Hubei Polytechnic University. His current research interest is computer network.

Zang Hui, he is a lecture in Computer School of Hubei Polytechnic University. His current research interest is data. mining.

Ni Bo, he is a PhD candidate in Computer School of Wuhan University. He is a teacher in Computer School of Hubei Polytechnic University. His current research interests include CSCW, CAD&CG, QOS in high speed network.

