Using Case-Based Reasoning to Support Web Service Composition*

Ruixing Cheng, Sen Su, Fangchun Yang, and Yong Li

State Key Lab. of Networking and Switching,
Beijing University of Posts and Telecommunications
{Chengruixing, liyong.bupt}@gmail.com,
{susen, fcyang}@bupt.edu.cn

Abstract. With the growing number of Web service, it is necessary to implement web service composition automatically. This paper presents an approach to support large-granularity web service composition accurately and fast according to users' requests. With this approach, it can reduce the cost of web service composition, and improve scalability, reusability and efficiency. Using a proactive well defined service base, web service composition execution engine can gain the logic with Case-Based Reasoning technology. Comparing to other method and qualitative analysis, the approach proposed by this paper can solve the problem of web service composition under the condition of insufficient and ill-defined knowledge, and can reduce the difficulty and cost of web service composition.

1 Introduction

With the emergence of numerous Web services, we need to, according to users' need, compose the current Web services from different environments, platforms and enterprises into a larger-granularity value-added service to satisfy uses' need. It remains an unsolved problem how to efficiently and precisely implement composition of services on a higher level.

This paper proposes a service composition method based on Case-Based Reasoning. First of all, this method builds the service base with structure of two layers, the basic service layer and application service layer, which is constructed according to the different service granularity. And by integrating and sealing cases in basic service layer into cases in application service layer with lager granularity. Then by comparing the similarity of different case, searching and reusing suitable case in the application service layer to obtain the description of composition logics, and realize larger- granularity service composition transparently in a simply, easy-to-use way. Furthermore, by reconfiguring users' individualized restrictive condition, this method can revise the case in the service base so that it can better satisfy users' need.

* This work is supported by the National Basic Research and Development Program (973 program) of China under Grant No. 2003CB314806; the National Natural Science Foundation project of China under Grant No.90204007; the National Natural Science Funds for Distinguished Young Scholar Program of China under Grant No.60125101; the program for Changjiang Scholars and Innovative Research Team in University (PCSIRT).

The second section of this paper introduces the related work. In the third section, the basic idea of using Case-Based Reasoning to support service composition is put forward. The fourth section introduces Case-Based Reasoning and the service-composition framework using Case-Based Reasoning. The last section gives a qualitative analysis of this method and illustrates the future work.

2 Related Work

The as-exist Web service composition methods can be classified into three kinds regarding whether manual tasks are involved in the composition process. They are manual composition method, half-automatic method and automatic method (see [1] for detail).

Manual method provides users interactive interface via graph or text editor. Users are involved in the produce of a script language in the service composition process. The language is then executed by service composition (see [2], [3] for detail).

Based on manual method, half-automatic method introduces the concept of semantic to reduce manual involvement, thus increasing the automatic level of service composition. Sirin and other scholars proposed an interactive service composition framework [4], which is based on semantic analysis. This framework can use semantic to perform filter and selection of service at different stages, produce and execute service composition work flow.

Automatic service composition does not need any manual interference in the whole service composition process. It normally adopts Artificial Intelligent method to realize the automation of the whole process. Among the related research results [5], there is a service composition framework based on Agent, using semantic and universal program to perform service composition. But to some of the service compositions, complicated negotiation and design are needed. The current Artificial Intelligent and work flow methods can hardly satisfy such need [6].

The current research work bases on the as-exist service composition template to realize dynamic service composition. But this method has a flaw in that the creation of service template needs manual interference. For instance, users have to design time constraints and non-function constraints. It also lacks the mechanism of learning from successful experience, making it highly dependent on professionals and experts. Therefore, this method cannot perform well when some users have special demand, or knowledge in certain area is unavailable, or certain domain has mal-definition.

3 Basic Ideas

This paper proposes a highly automated service composition method with large-granularity. The basic idea of this method is to apply Case-Based Reasoning to the stage of service composition discovery, while moving the logic design of the existing service composition engine to Case-Based Reasoning system. This can be done through searching and modifying case. With the well designed definition predetermined by area experts and the reuse of users' past successful experience, this method can establish all sorts of service composition logic set for users before they request. In the service composition process, the method can, according to users requests, search，revise cases, procure satisfactory service composition logic sets and give them to the service execution engine to process.

The structure of service base can be divided into two layers to reduce the complexity of the service composition and to improve the reusability and maintainability. The first layer is basic service layer, which is related with specific domain. The domain experts design case in basic service layer with the interior logic between the services in the domain, and establish service specification and basic infrastructure to map specific physical service to specific domain case. It will not be discussed how to realize the virtualization from physical service to specific domain case in this paper. The second layer is application service layer. By setting some individualized constraints, users can use the services provided by the first layer directly to design application without worrying about how the services in a domain are combined. In this way, users are freed from the complicated service composition process. What they need to do is simply the process of case matching and configuration. The case in the application layer is preconfigured by experts. It can also be a composition of case from a variety of domains based on past successful service composition experience allowing the dynamic configuration of some individualized constraints. An application layer after dynamic configuration can describe a complete application. When a user's request arrives to the application layer of the service base, the suitable case in the application service layer can be found by the process of searching and matching. If no suitable case is found, the pre-constraints will be partly modified and an application description that can satisfy user's need will be formed. The user will evaluate the result from the execution engine. If it is satisfactory, the service base will save the modified case.

When the number application layer case grows into a large one, to increase the efficiency of searching, we first cluster cases in the application service layer. Through clustering, the application service layer cases are divided into different categories. Then we compare user's request with cluster centre, locate the certain category it falls into according to its similarity to cluster centre case, and perform search and match in the very category.

4 Service Composition with Case-Based Reasoning

4.1 Case-Based Reasoning (CBR) and the Design of Web Service Composition

CBR originated from human being's cognition activity. It is a kind of analogism [7]. CBR's basic idea is to form new solutions to new problems and new cases through certain analogy and through appropriate adjustment according to the difference between old problems and new ones. The analogy is made based on the experience and knowledge people have acquired from the past activities dealing with similar problems. To construct service cases base system, sufficient experience of service composition is a must. Long accumulated experience from domain experts and knowledge engineers is needed. On the other hand, direct acquisition of large amount of useful data from past successful experience is also necessary. SB stores substantive service composition cases. To better maintenance and reduce the complexity of service composition, according to the service case granularity, the case base can be divided into two layers: basic service layer and application service layer.

Cases in basic service layer are certain standards summarized based on the analysis of domain experts according to the requirements in the domain, the specific background and the past successful experience. According to these standards and optimized rules, Cases in basic service layer abstract and encapsulate the service logic and composition logic in the domain, and map many atomy service to a Case in basic service layer by service virtualization mechanism.

An Application service layer case consists of one or many basic service layer cases, and can describe composition logics according to successful experience. It can express a larger granularity service.

4.2 Similarity Comparison of Cases

To estimate the similarity between the current case and the new problem, according to the comparison of the cases' similarity, is the key of the case based reasoning and also the essence of showing the correctness of reasoning and the intelligent of performance. By comparing the interface information of cases, the case with the biggest similarity is retrieved. Then the case will be modified according to user's constraints. In the process of case retrieval, to improve the efficiency, we first perform clustering to the case in the case base. Then we decide which kind it belongs to by making similarity comparisons with the case from cluster centre. The algorithms to perform clustering will not be discussed in this paper. The case similarity calculation form adopts attributes-value mode to calculate the similarity in the prototype. The selection of feature weight is significant to CBR system. A weight value of an attribute reflex not only its importance compared to other attributes, but also the level of contribution it will make to problem solving. In prototype, the interface information in the case is for case retrieval. The name of the base case in the interface information in the application layer is essential for problem solving. Therefore the name of a base layer case will be rendered higher weight. The constraint reflexes users' individualized requirement, which often change with the command of different users. Therefore it is rendered lower weight. But when users are very sensitive to a certain constraint, it can also be added higher weight. For example, if a user has a strict requirement to price, it can be given higher weight. CBR system judges to which attributes users are sensitive according to both the past successful experience and the constraints users have input, and renders them corresponding weight. Besides, different attributes have different contribution in solving problems, thus should be rendered different weight value. The selection of algorithms dealing with weight is really important to the optimization of CBR system.

This paper uses a method that performs case retrieval by comparing the parameters in interface information in a case. These parameters can be divided into two kinds - text parameters and data ones. To text parameters, comparisons will be made through the algorithm related with the similarity between concepts. To data parameter, the scope of threshold will be made. The similarity will be zero if the threshold is not in the scope. Otherwise, it will be between 0 and 1.

The algorithm is described below taking service case base in the application layer as an example.

Suppose that the service case base in the application layer has N cases:

$$CBA=\{AC_1,AC_2,\ldots,AC_n\}$$

There are m parameters in the interface information in each application layer case (AC). There parameters' weight vector are $W(W_1,W_2,W_3,\ldots,W_m)$,并且 $\sum_{i=1}^{m} w_i = 1$

The similarity of Case CA_x and Ca_y is: $S(x,y)=1-\sqrt{\sum_{i=1}^{m} w_i \times d(CA_{x_i},CA_{y_i})^2}$.

$\sqrt{\sum_{i=1}^{m} w_i \times d(CA_{x_i},CA_{y_i})^2}$ is the Euclidean distance between CA_x and CA_y, $d(CA_{x_i},CA_{y_i})$ is attribute normal distance.

1) When the type of the parameters is text:
If the text keywords are the same, $d(CA_{x_i},CA_{y_i}) = 0$. Otherwise $d(CA_{x_i},CA_{y_i}) = 1$.

2) When the type of the parameters is data: $d(CA_{x_i},CA_{y_i}) = \frac{|CA_{x_i}+CA_{y_i}|}{|T_{\max}-T_{\min}|}$

$|T_{\max} - T_{\min}|$ Represents the scope threshold

4.3 The Framework of Service Composition with CBR

The framework of service composition with CBR is shown in figure 1. It can be divided into two parts. The first one realizes this module's ulterior management interface, controlled by domain expert. By managing the view, the experts can manage and build the corresponding service base according to different domain. In addition, they can also test the correctness of the service cases by formal validation tools. Each CBR processor will store local service base and the service base address list scattered in other nodes. The second part deals with users' command. The whole process, from receiving a user's request to producing the result, can be divided into five steps.

1) The user put a request to the user request processor;
2) The processor first divides the request into request parameters and constraints, then passes the result to CBR processor;
3) The CBR processor classifies the user's request according to the request parameters and constrains. To improve the preciseness of the classification, we use a method that supports vector machine. The related work has been stated in Section three. According to the result we search the related service case base to obtain the most similar source case. Then, we can get appropriate service composition description by making comparisons with the source case in similarity. A detailed explanation has been made in the former part of this section.
4) CBR processor translates the description into control logic relations according to the service composition cases found and the user's individualized constraints;
5) Service composition processor takes actions according to the control logic relations;
6) If the user is satisfied with this service, the logic relations will be saved as application case in the service case base. The domain experts will, according to users' different response, fill in corresponding service records, modify and make complements to the service case base.

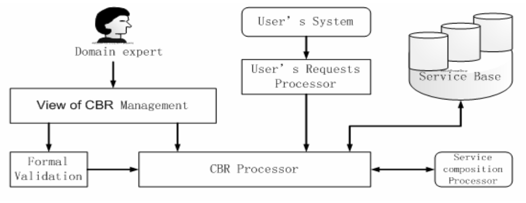

Fig. 1. The Framework of service composition with CBR

5 Experiment and Analysis

5.1 Simulation Experiment

In order to stand the content out, the experiments are based on the hypothesis as follows:

1) All requests can match with cases in the application service layer
2) The time of executing logics will not be taken into account.
3) The time of transmission in the net work will not be considered.

Exp 1. Test of service composition efficiency.

First, 100 cases from the application service layer are selected at random. Then different experiments are made according to the number of sub-services contained by case in application service layer. Comparisons will be made between CBR and the method based on keyword (KC) or semantic (SC).

Fig.2 shows the mean time of service composition of three different method of web service composition. It is shown that the relationship between the number of service and the time spent by the processing of service composition. In addition, it can be shown that the use of the service composition method with Case-Based Reasoning (CBR) can make the efficiency of web service composition higher than that of method based on key word (KC) or semantic (SC).

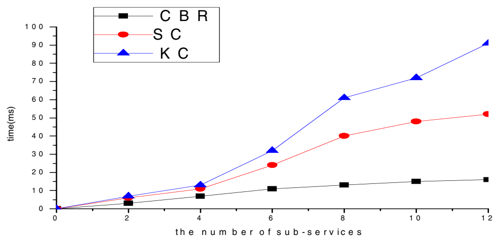

Fig. 2. Experimental results about service composition efficiency, varying the value of three different methods

Exp 2. Test of successful service composition percentage.

First, 100 requests whose similarity is 70%,75%,80%,85%,90%,95%,100% are created respectively. That is, totally 700 request are established. Then comparison are

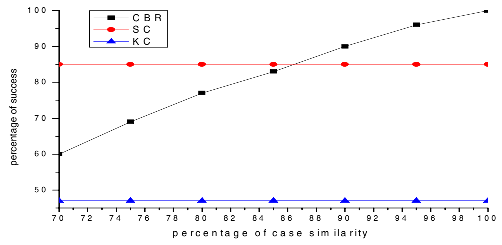

Fig. 3. Experimental results about successful service composition percentage, varying the value of three different methods

made between the Case-Based Reasoning (CBR) method and the method based on key word (KC) or semantic(SC).

Fig.3 shows the mean successful percentage of three different method of web service composition. It can be shown that when the similarity of a case increases, the percentage of successful composition also increases. Under the condition that sample number is 100 and case similarity is greater than 85%, using the service composition method with Case-Based Reasoning (CBR) can make the percentage of success greater than that of the method based on keyword (KC) or semantic(SC).

5.2 Qualitative Analysis and Conclusion

We present a method of using Case-Based Reasoning (CBR) to support web service composition. CBR is applied in the phase of pre-composition to build service based of different service domain. When user's request is received, the process of search and matching cases of application layer in service base is called. And the list of sub-services, parameters and condition of restriction are gained according to the source case. Then they are converted into control logics of web service composition by CBR processor and finished by web service composition processor.

Compare to other methods of web service composition, this method have many advantages shown as follow:

1) Case-Based Reasoning is applied in the process of building service base, before service composition engine deals with user's requests. By searching in the service base, the logic of web service composition is discovered. The result can avoid starting from scratch in designing web service composition logics. It can also simplify the process of reasoning the logic, and improve the efficiency of service composition engine, as well as reduce the cost of service composition.
2) Compare to the method of service composition based on template, ours doesn't need interference of users or experts during the process of composition. It can reuse the successful experience of composition. In addition, it can overcome the difficulties of designing suitable logic of composition under the condition of insufficient knowledge or mal-defined knowledge. So it improves the reliability of applications.
3) With the times of successful service composition increasing, the cases suitable for user's requests in service base also increase. Furthermore, because of applying clustering method in the process of matching and search suitable case in service base, the efficiency of discovering source cases is greatly improved.

4) Cases in service base are all designed by domain experts and validated ,or preconfigured according to successful experience, so it guarantee the correctness.
5) By applying Case-Based Reasoning to web service composition, the service base can be revised according to service feedback. This mechanism improves the ability to learn and reduces the dependence on users or experts.

6 Future Work

There are many related work to be perfected at present. In the future, we intend to take steps to (1) integrate induce algorithm into our method of web service composition with Case-Based Reasoning to improve the ability to learn, (2) realize the virtualization mechanism between physical layer and basic service layer, (3) improve the clustering and similarity algorithm, and (4) add the mechanism of QoS evaluation to our method and use the algorithm based on index of QoS or other recommendations in the process of searching cases in the application service layer.

References

[1] Schahram Dustdar, Wolfgang Schreiner. A survey on web service composition[J]. Web and Grid Service,Vol.1 No.1,2005
[2] Mandell D. J . , McIlrait h S. A. . A bottom up approach to automating Web service discovery , customization , and semantic translation. In : Proceedings of t he 12th International WWW Conference Workshop on E-Services and the Semantic Web ,Budapest , 2003 ,89～96
[3] Taylor I. , Shields M. , Wang I. , Philp R. . Distributed P2P computing within triana : A galaxy visualization test case. In :Proceedings of t he IPDPS 2003 Conference , Nice , France ,2003 , 16～27
[4] E Sirin, B Parsia, J Hendler .Filtering and selecting semantic Web services with interactive composition techniques - IEEE Intelligent Systems, 2004 - ieeexplore.ieee.org
[5] McIlraith S. , Son T. C. . Adapting golog for composition of semantic Web services. In : Proceedings of the 8th International Conference on Knowledge Representation and Reasoning , Toulouse , 2002 , 482～493
[6] J. Koehler and B. Srivastava .Web Service Composition: Current Solutions and Open Problems. ICAPS(The International Conference on Automated Planning & Scheduling) 2003 Workshop on Planning for Web Services
[7] Mario Lenz. Case-based Reasoning: From Foundations to Applications [M]. Berlin: Springer, 1998