2008 International Conference on Computer Science and Software Engineering

Web Services Composition Method Based on OWL

Jike Ge Yuhui Qiu* Shiqun Yin
Faculty of Computer and Information Science
Southwest University
Chongqing, China
{gjkid, yhqiu, qiongyin}@swu.edu.cn

Abstract—**At present, Web services are created and updated on the fly. It has already beyond the human ability to analysis them and generate the composition plan manually. It is a problem that composing existing Web services automatically and dynamically according to users' request. A number of approaches have been proposed to tackle that problem. Most of them are inspired by the researches in cross-enterprise workflow and AI planning. In this paper, we propose a Web services composition method based on OWL ontology, and design a system model for services composition. Web services are modeled based on OWL ontology, the services are semantically matched and composed, and the executing plan is generated. Finally, the plan is executed and valuable results are returned to users. We also provide the experimental comparison, and report that our method has more accurate matching results.**

Keywords: Web service; ontology; matching; composition

I. Introduction

Web services are considered as self-contained, self-describing, modular applications that can be published, located, and invoked across the Web[1]. Under the developing of the Internet, a large number of Web services have emerged with an increasing amount of organizations only implement their core techniques and outsource other application services over Internet. However, single service published on the Web often can not satisfy users' request. Therefore, the ability to select and compose inter-organizational and heterogeneous services on the Web efficiently and effectively is an important step towards the development of the Web services application. Web services automatic composition technology become one of the main concerns of the application development process[2].

In the research related to Web services composition, several methods have been provided that will allow easy integration of heterogeneous systems. Such as, Universal Description, Discovery and Integration (UDDI)[3], Web Service Description Language (WSDL)[4], Business Process Execution Language for Web Service (BPEL4WS)[5], which are focused on representing services compositions where flow of a process and bindings between services are known a priori. Despite all these efforts, the Web service composition still is a highly complex task, and it is already beyond the human capability to deal with the whole process manually. The complexity, in general, comes from the following sources. First, the number of services available over the Web increases dramatically during the recent years, and one can expect to have a huge Web service repository to be searched. Second, Web services can be created and updated on the fly, thus the composition system needs to detect the updating at runtime and the decision should be made based on the up to date information. Third, Web services can be developed by different organizations, which use different concept models to describe the services, however, there is not a unique language to define and evaluate the Web services in an identical means.

Semantic Web is the key step to Web services composition. The functionality of a Web service needs to be described with additional information, either by a semantic annotation of what it does or by a functional annotation of how it behaves. The semantic Web is also an extension of the current Web in which information is given well defined meaning, consequently better enabling computer and human to work in cooperation. Semantic Web aims to add machine-interpretable information to Web content in order to provide intelligent access to heterogeneous and distributed information[6]. But, many Web services can not well support semantic service description. Therefore, some researchers are now using ontology to help capture Web service semantics.

In this paper, we extend WSDL with semantic capabilities for semantic Web services. This makes it feasible for the automatic Web services composition. We define ontology for Web services and specify it using the Web Ontology Language (OWL)[7]. By specifying Web services, we get services composition plan. Our method analyzes the structure of services and makes the matched services more accurately.

The rest of this paper is organized as follows. Section 2 introduces the definition of Web services and related concepts based on OWL. Section 3 describes the Web services composition model based on OWL, it is our main contribution. Section 4 reports the experimental analysis and their evaluation. The last section concludes the paper.

II. Web Services Description

A. *Web Service Definition*

Composing Web services requires the description of each service so as to other services and users can understand its features and then interact with it. It maybe occur the problem of semantic conflicts during composing Web services for these Web services may come from different domains. In order to solve this problem, we define the Web services and related concepts, and specify it using OWL.

* Corresponding Author

Definition 1 (Web Service): Given a Web Service (WS), WS=<Service-name, Description, Ops, IN, OUT, Binding, Domain>.

Where, **Service-name** is the name of a Web service; **Description** is a text summary about Web service; **Ops** is the set of actions supported by the service; **IN** and **OUT** are the set of Web service' inputs and outputs; **Bindings** is the set of binding protocols supported by Web service, such as UDDI, WSDL, RDF; **Domain** denotes the category of Web service.

Definition 2 (Ops): Given an action of the Web service (Op_i), Ops are the set of Op_i. Op_i=<name, description, catalog, in, out>

Where, i={1,2,3,…,n}, n=|actions|; **name** is the name of Op_i, **description** is a text summary about Op_i; **in** and **out** are the set of Op_i's inputs and outputs; **catalog** indicates the category of Op_i.

Definition 3 (catalog): Given a catalog of the Op_i, catalog= <categoryName, taxonomy, value>

Where, **categoryName** is the name of the actual category, which could be just a literal or a property; **Taxonomy** stores a reference to the taxonomy scheme, it can be either a URI of the taxonomy or a URL where the taxonomy resides, or the name of the taxonomy; **value** are the values in a specific taxonomy.

B. The Rules of Web Services Composition

In general, services composition can divide into catalog composition and binding composition[8]. When invoking a composed service, it should be ensure that sub-services in the service matching well. Otherwise, it would be difficult to invoke an operation if there were no matching information between the parameters requested by this operation. So, we should define the rule of services composition explicitly.

Rule 1 (catalog composition): Given two Ops, Op1 and Op2, Op1.catalog compositeWith Op2.catalog to be tenable, if and only if the following conditions are true:

- Op1.catalog. categoryName= Op2.catalog. categoryName
- Op1.catalog.taxonomy=Op2.catalog.taxonomy.

For example, assume two Web services are communicating through operations. These Web services may support different binding protocols (SOAP, HTTP, et al), it is important to insure that they understand each other at the protocol level, or one of the protocols adopted by one Web service must be supported by the other. The rule 1 can insure this condition is true.

Rule 2 (Binding composition): Given two Web services, WS_1 and WS_2, WS_1.Binding compositeWith WS_2.Binding to be tenable, if and only if $WS_1 .Binding \cap WS_2 .Binding \neq \phi$.

Rule 3 (operation sequence): Pre(Op_1, Op_2), denotes Op_1 precedes op_2, to be tenable, if and only if Op_1 and Op_2 satisfy the following rules:

- $Op_1.out \supseteq Op_2.in$
- Op_1.catalog compositeWith Op_2.catalog
- WS_1.Binding compositeWith WS_2.Binding

The above rules ensure services composition success in theoretical.

III. WEB SERVICES COMPOSITION BASED ON OWL

A. The Framework of Web Services Composition

The framework of Web services composition in Fig. 1 consists of service requester, execution module and services composition matchmaker.

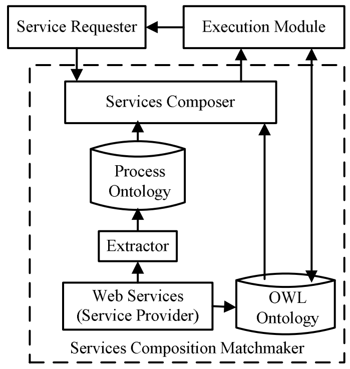

Figure 1. The architecture of services composition

First, Web services, which provided by the service provider, need to be registered for composing process. The service provider registers its service semantic in the OWL Ontology. Then, the extractor extracts the necessary contact details, such as, service name, service description, instance of relationship, input/output information, etc, and stores them in the process ontology. The process ontology is also updated by connecting the I/O parameters of the new service to other compatible service parameters, which can reduce the complexity of searching for services in an automatic composition. The services composer searches for a sequence of services and matches appropriate services, it can return a composite service with the optimal graph, and generates composition plan of the composite service for the execution module. Finally, the execution module executes plan and calls related Web services, and sends the final results to the service requester.

B. Service Matching Algorithm

Service discovery, often referred to as service matching, is the precondition of services composition. But, in Web and semantic Web scenario, heterogeneity cannot be avoided. Different actors have different interests and habits, use different tools and knowledge, and most often, at different levels of detail. These various reasons for heterogeneity lead to diverse forms of heterogeneity. Therefore, during service discovery, we should be carefully taken into consideration.

Because of the limitation of syntactic discovery, there are several efforts in the area of semantic web service discovery[9], especially Ontology Web Language for Services (OWL-S) and Web Services Semantics--WSDL-S. Rich semantic description of services is important to facilitate semantic discovery. The description of services should reflect their functional characteristic, e.g. by trying to describe the function of a service with a single term. This kind of semantic description can be found in [10]. This method ignores the operational characteristic of services. The matching for single input and output concepts is put forward in[11]. In this paper, we propose a novel Web service matching approach that allows for more flexible and useful description. The Web service matching algorithm shows in Fig. 2.

```
Algorithm: Matchmaking (IN, $OUT_i$) // IN denotes the inputs of the request, $OUT_i$ denotes an output of the desired service
{
    Get all the $op_j$ and IN $\supseteq op_j$.in
    Add $op_j$ to set B
  Repeat
  {
  For each the new added $op_j$ in set B
        Get all the $op_i$ and Pre ($op_i$, $op_j$)
  } until B does not increase
 Get all the output of B and add it to set A
  If (A $\supseteq OUT_i$)
     Return true;
  Else
     Return false;
}
If $OUT_i$ can be satisfied by the registered services and the input set, we will find the services and store them in directed graph (G).

Find (IN, $OUT_i$ )
{
  if (Matchmaking (IN, $OUT_i$ ))
  {
    for all the registered WS
       A=get (Ops, $OUT_i$ ) // get all the Ops whose output include $OUT_i$
   Add $A_i \in$ A to G as the pre node of the $OUT_i$
   For all $A_i \in$ A
   { if (IN $\supseteq A_i$ .in )
      {
      for  each $in_i \in A_i$.in and $\not\subset$ IN
       {
        Find(IN, $in_i$)
       }
      }
     Return G;
   }
  }
```

Figure 2. The service matching algorithm

There are four cases for check similarity of an output and input parameter from the same ontology:

- They are the same, their similarity is maximal.
- The output parameter of the former service is subsumed by the input parameter of the later service. They are the second best matching, and the similarity value depends on their distance in the ontology.
- The output parameter of the former service subsumes the input parameters of the later service, and the properties of the parameters could be partially satisfied.
- Two parameters have no subsumption relation or they are come from different ontology, the similarity value can be obtained by Tversky's feature-based similarity model [12], which is based on the idea that common features increase the similarity of two concepts, while feature difference decreases the similarity.

Finally, we could look for the services for each of the outputs in OUT with the service matching algorithm.

C. *Web Services Execution*

According to the results of the above algorithm, we can generate a detailed description of a composite service. In order to execute the composition plan automatically, we need to convert **G** to BPEL4WS data flow, this can be done by the services composer and produce the BPEL4WS code according the BPEL4WS syntax specification. After the BPEL4WS is created, the execution module will execute the BPEL4WS and send the results to the service requester.

IV. Experimental Evaluation

A. *Evaluation Set-up*

We present the performance and quality evaluation of our proposed method with other methods for Web services composition. In the Web services composition methods, some matchmaking algorithms based on keywords, we call them KW[13], have been proposed. But, they do not take the semantic between services into consideration so that they have some limitations on the quality of composition. Another, we call them SM, has been well researched in[10], which add semantic to Web service but do not take the structure of the Web service into consideration. Our Web services composition method based on OWL, we called OC, which considers not only the semantic but the structure between services.

All the tests have been performed on a Intel Pentium Dual 1.6GHz, with 1 GB of RAM, with the Windows XP operating system.

B. *Precision Evaluation*

We compared precision against KW and SM. We random generate 200 requests, and respectively use **N**=1000, 1800, 2500 registered Web services. We use the following equation as measure method:

$$P = K / N \qquad (1)$$

Where, **P** indicates the proportion of request that can be successfully fulfilled, **K** is the number of request that can be successfully fulfilled in **N**. **N** is the number of registered Web service. Fig. 3 shows the comparison results.

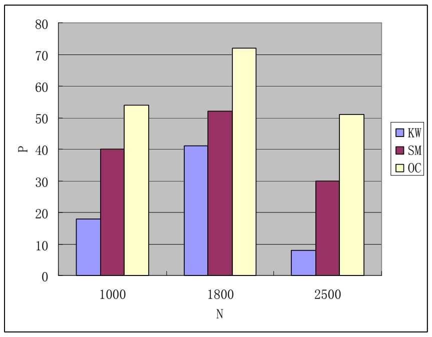

Figure 3. The precision of Web services composition

In Fig. 3, we can see that the **P** of SM and OC is much higher than that of KW. The method based on keywords can only support querying which may bring about low precision results, and it is difficult to discovery semantic related requests. Our method is a bit more precise than that of SM because the SM neglects the process of service composition.

C. The QoS Evaluation

Quality of service (QoS) is the ability to provide different priority to different applications, users, or data flows. It has become an important feature in evaluating system performance, and it is also a criterion of users' satisfaction degree.

In this experiment, we will test the QoS of three methods. We randomly generate 8 requests and use N=1000 registered Web service to test the users' satisfaction degree. We define $\sigma=0.5$ as a reference standard value that indicates the base line at which the users can accept the results of the Web services composition.

As shows in Table I, KW has the best results in the users' satisfaction degree, but it is so perfect that the results cannot be believed by people. Maybe the results of KW fit for all requests, the results are not pertinence, and they aren't the best answers. The results of OC are better than SM. It is proved that our proposed method has a better performance than other methods.

TABLE I. THE USERS SATISFACTION DEGREE OF WEB SERVICES

Requested service	KW	SM	OC
1	1	0.83	0.96
2	1	0.75	0.85
3	1	0.65	0.83
4	1	0.87	0.96
5	1	0.91	0.90
6	1	0.85	1
7	1	1	1
8	1	0.85	0.83

From above evaluation, our proposed method has a better performance than other two methods. It is proved that our method can ensure quality and efficiency while composing services and it is more satisfied with users' requests.

V. CONCLUSIONS

In this paper, we introduce an automatic Web services composition approach based on OWL ontology, and propose a service matching algorithm. Our method can compose the services which described by OWL efficiently according to user's request. The experiment results also proved our method has a better performance than other methods.

Even though services composition is a well recognized problem. But, the presented composition strategies are almost sequential, and a composite service might require a few services to be executed in parallel. It might be possible to enhance our method to start parallel service matching to handle parallel service flows in the services composition. We plan to continue our research in this direction to realize more efficient parallel services composition methods.

REFERENCES

[1] J. Rao, and X. Su, "A Survey of Automated Web Service Composition Methods," In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC'2004, LNCS, San Diego, USA, Springer-Verlag, pp. 43-54, July 2004.

[2] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana, "The next step in Web services," Communications of the ACM, vol. 46, no. 10, pp. 29-34, 2003.

[3] T. Bellwood, S. Capell, J. Colgrave, et al, "Universal Description, Discovery and Integration specification (UDDI) 3.0.2," Online: http://uddi.org/pubs/, October 2004.

[4] R. Chinnici, J. Moreau, A. Ryman, et al,"Web Services Description Language (WSDL) Version 2.0,"Online: http://www.w3.org/TR/wsdl20/, June 2007.

[5] T. Andrews, F. Curbera, H. Dholakia, et al, "Business Process Execution Language for Web Services (BPEL4WS) version 1.1," Online: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel, May 2003.

[6] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American, vol. 284, no. 5, pp.34-43, May 2001.

[7] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, "Web Ontology Language (OWL) Abstract Syntax and Semantics," Technical report, W3C Recommendation, 2004.

[8] M. Dumas, J. O'Sullivan, M. Heravizadeh, D. Edmond, and A. Hofstede, "Towards a semantic framework for service description," In Proceeding of IFIP Conference on Database Semantics, pp. 277-282, 2001.

[9] J. Euzenat, and P. Shvaiko, Ontology Matching, Springer Berlin Heidelberg, New York, 2007.

[10] M. Paolucci, T. Kawmura, T.R. Payne, and K. Sycara, "Semantic matching of web services capabilities," In Proceeding of 1st International Semantic Web Conference, LNCS, Berlin, Heidelberg, Springer-Verlag, pp.333-347, 2002.

[11] B. Benatallah, Q.Z. Sheng, and M. Dumas, "The self-serv environment for Web services composition," IEEE Internet Computing, vol. 7, no. 1, pp. 40-48, January/February 2003.

[12] A. Tverski, "Features of similarity," Psychological Review, vol. 84, no. 2, pp. 327-352, 1977.

[13] A. Sajjanhar, J. Hou, and Y. Zhang, "Algorithm for Web Services Matching," In Proceedings of APWeb2004, LNCS, Berlin Heidelberg, Springer Verlag, pp. 665-670, 2004.