A Framework of Web Services Discovery and Composition Based on Semantic

Zhiling Huang[1] **Weihua Ai**[2]
[1]The information and technology center of Yi li Technical and Vocational College, Yining, Xinjiang
[2]College of Meteorology and Oceanography, PLA University of Science and Technology, Nanjing 211101, China
awh1979@126.com

Abstract - A framework of web services discovery and composition based on semantic is proposed to find appropriate web services according to the requests of user. Some service discovery and composition algorithms are used in this framework. The component of service description can transfer the services requirements of the user to semantic description. When a formal service request is defined, the service discovery component queries the service repository for a service that matches the service request. In case no match is found, the composition component creates a composite service that matches the request with optimal QoS.

Index Terms - Web services; service composition; quality of service; knapsack problem

1. Introduction

Web services technologies[1] provide a suitable technical foundation for developing and deploying loosely coupled and reusable software components, which can be invoked through their service ports. The continuous growth in the Web services can somewhat meet the increasing needs for the services, but the available services can not always satisfy the needs of requesters. Therefore Web research attempts to provide complex services by, in effect, combining simple services in the way of a workflow of services, and such combination is called service composition. In order to tackle the challenge of service composition, most of the work done until now has focused on two main composition approaches, namely by considering function and quality of services. The approach based on functional aspects aims at finding a sequence of atomic components described in terms of their IOPEs that matches a given request. Web services composition thus creates a quality of service (QoS) engineering problem since the services selection must select the best services to compose an efficient complex service with QoS assurance. The aim of service composition based on QoS is to search for the optimal set of services that, composed to create a new service, result in the best QoS, under the user or service designer constraints. One may decide to choose the cheapest service, the fastest, the most reliable or maybe a compromise among them.

In this paper, we focus on the study of services discovery and composition framework considering the services discovery and composition algorithms which are based on semantic. The outline for this paper is as follows. The section 2 presents the framework of services discovery and composition and introduces the main function of each component. Section 3 introduces the service discovery algorithm. Section 4 introduces automatic web services composition method based on semantic. Section 4 presents the composition algorithm based on QoS. Section 4 and 5 list the experimental and evaluation results. The last section concludes the paper.

2. Framework of Web services composition

The main concern in creating this framework was to create an interface that can interact with the user and transfer user language commands into a logic based language that could allow reasoning. Finding appropriate web services according to the ontological domain of the problem was also the aim of our framework. The proposed framework consists of four main parts: services description, services composition, services discovery, services and Ontology database, as shown in figure 1.

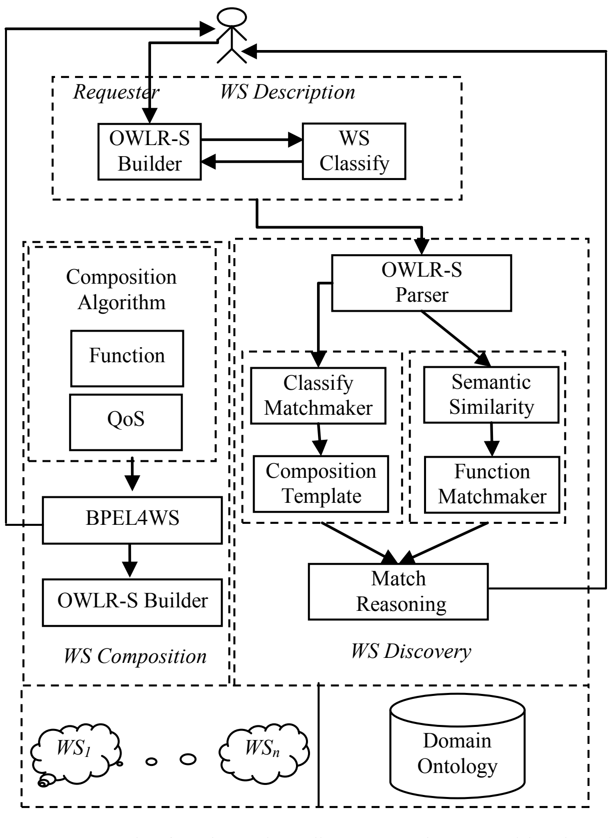

Fig.1 Framework of Web services discovery and composition based on Semantic

The Web services description is a component which has the most interaction with the user which transfer the services requirements of the user to semantic description The requirements with semantic are the inputs for service discovery and composition components. The services and Ontology database component is used to store the services and domain Ontology registered. The services discovery and composition component are the core of the framework. When a formal service request is defined, the service discovery and composition component queries the service repository for a service that matches the service request. If a match exists on the repository, the matching service or composition template is returned. In case no match is found, the composition component creates a composite service that matches the request. In principle, the composition component may generate multiple alternative compositions that match a service request with optimal QoS.

3. SERVICE DISCOVERY

Paolucci[2][3] proposed a service matchmaking approach based on OWL-S[4]. They make use of service profile section of OWL-S to describe the IOPEs of a service. Their matching algorithm works as follow: first, they compare the outputs of the request service with all outputs of existing advertisement services, a match is recognized if and only if for each output of the request, there is a matching in the advertisement. Else, the match fails. For inputs, the order of the request and the advertisement is reversed. To achieve flexible matching, they define five degree of service matching: Exact, Plug-In: Subsume, Intersection, Not Relevant. Degrees of the match are organized in a discrete scale. Exact matches are clearly preferable; Plug-In matches are considered the next best; Subsume matches are considered to be third best; Intersection is considered to be fourth best; and Not Relevant is the lowest level, since it is a failed match.

In this paper, we compare the function parameters such as input and output from profile of requirement and advertisement. We compute similarity of the two service profile based on service matching pair[5]. The service matching graph defined is a graph which represents the relationship among service parameter, service parameter pair and service matching pair. The matching problem may be transformed into a weighted bipartite matching problem as show in figure 2.

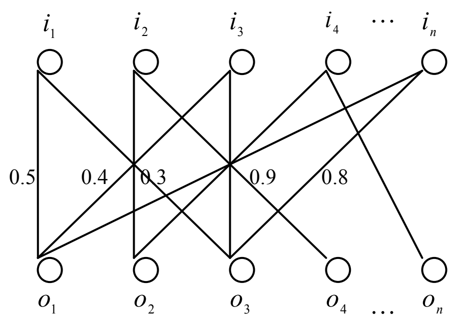

Fig.2 Perfect Matching

Assuming the outputs of service WS_m can be the inputs of the service WS_n in the service composition. We define $SA(WS_m, WS_n)$ the services association degree between WS_m and WS_n, the measure of which is reduced to compute the similarity between $O_m(o_1, o_2, \cdots, o_m)$ and $I_n(i_1, i_2, \cdots, i_n)$, where $O_m(o_1, o_2, \cdots, o_m)$ is the output parameters of service WS_m and $I_n(i_1, i_2, \cdots, i_n)$ is the input parameters of service WS_n. The similarity measure of the two concept set is an unrepeatable assignment problem with best time complexity. Suppose $G_{m,n} = (O_m, I_n, E)$ is an undirected weighted bipartite, where $O_m = (o_1, o_2, \cdots, o_m)$ and $I_n = (i_1, i_2, \cdots, i_n)$, and each edge linking o_i and i_j represents the similarity of the parameter pair $SPP_{i,j}$. The measure of services association degree is reduced to problem of computing the perfect matching in the bipartite graph. We consider the problem of finding a maximum weight perfect matching in the service bipartite graph $G_{m,n} = (O_m, I_n, E)$ to get the maximum weight matching, where O_m is the set of output parameters in service WS_m and I_n is the set of input parameters in service WS_n. We use the improved Kuhn-Munkres[6] algorithm to compute the service association degree between the two services. The Kuhn–Munkres Algorithm: Start with an arbitrary feasible vertex labeling and choose an arbitrary matching M in $G_{m,n}{}^{/} = (O_m, I_n, E_l)$.

4. SERVICE COMPOSITION FOR FUNCTION

In case no match is found by services discovery component, the composition factory is used to create a composite service that matches the request. In principle, the composition component may generate several service compositions that can match a service request. Here an automatic service composition for function is used to return services composition sequence which can satisfy requested service's needs according to the existed services[7]. In the method services association degree is the base for service composition and we use the improved Kuhn-Munkres algorithm to get the degree. Let $WS = (WS_1, WS_2 \cdots, WS_n)$ be a set of Web services in the service community. λ and T are the service association degree's threshold and QoS threshold respectively. The effects of the two thresholds are to control the service composition's quality and to save the composition time. $WSR_r(I_r, O_r)$ is the requested service and service composition sequence is WSQ (Web Services Queue). The composition algorithm considers the possibility to accomplish the goal directly with a single service. Then if it is not possible to satisfy the request with a single service the method verify the possibility to match the request by composing various services.

The implemented tools for this composition algorithm include Eclipse, OWL-S API[8] and Jena API[9]. The inputs and outputs parameters of all the services are expressed by OWL-S and domain ontology. We ran our experiments on a Windows machine with a Pentium Centrino 2.4 GHz CPU and 768MB memory. The service composition can find all possible *WSQ* and rank them with the value of QoC. But finding all the possible *WSQ* is time consuming. Figure 3 shows the composition time increases with the service number. Figure 4 illuminate the relationship between service composition time and service association degree's threshold λ for different number of service where QoC=0.5.

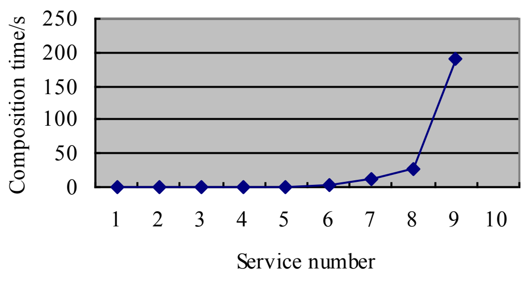

Fig 3. Composition time increases with the service number

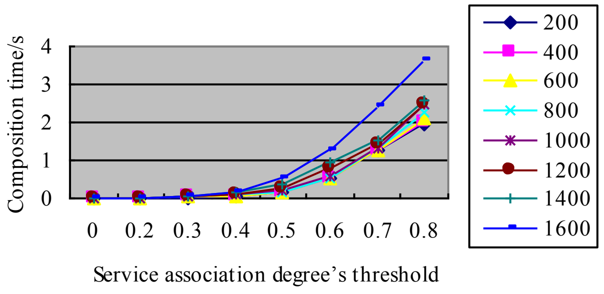

Fig4. Composition time with QoC=0.5

5. Service Composition based on QOS

The Composition component may generate multiple alternative compositions that match a service request. The best services composition should be selected with optimal QoS[10]. Here assumed given a task t_i in a composite service, there is a set of candidate Web services $S_i = \{s_{i1}, s_{i2}, \cdots, s_{ij}\}$ that can be used to execute this task. In other words the set of all candidate services may be available with the same functions, however they surely be different Quality of Service. By merging the quality vectors of all these candidate Web services, a matrix Q showed below is built.

$$Q = \left(Q_{i,j}^k ;1 \le i \le n,\ 1 \le j \le l_i,\ 1 \le k \le m\right) \qquad (1)$$

In the matrix, each row Q_i corresponds to a set of candidate Web services, and Q_{ij} corresponds to the service s_{ij}, while k expresses a quality dimension. Some of the criteria could be negative, i.e., the higher the value, the lower the quality. This includes criteria such as execution time and execution price. Other criteria are positive, i.e., the higher the value, the higher the quality. For negative criteria, values are scaled according to (2). For positive criteria, values are scaled according to (3).

$$V(Q_{i,j}^k) = \begin{cases} \dfrac{Q_{i,j}^k - \min\limits_{v=1}^{l_i}(Q_{i,v}^k)}{\max\limits_{v=1}^{l_i}(Q_{i,v}^k) - \min\limits_{v=1}^{l_i}(Q_{i,v}^k)} & \max\limits_{v=1}^{l_i}(Q_{i,v}^k) - \min\limits_{v=1}^{l_i}(Q_{i,v}^k) \neq 0 \\ 1 & \max\limits_{v=1}^{l_i}(Q_{i,v}^k) - \min\limits_{v=1}^{l_i}(Q_{i,v}^k) = 0 \end{cases} \qquad (2)$$

$$V(Q_{i,j}^k) = \begin{cases} \dfrac{\max\limits_{v=1}^{l_i}(Q_{i,v}^k) - Q_{i,j}^k}{\max\limits_{v=1}^{l_i}(Q_{i,v}^k) - \min\limits_{v=1}^{l_i}(Q_{i,v}^k)} & \max\limits_{v=1}^{l_i}(Q_{i,v}^k) - \min\limits_{v=1}^{l_i}(Q_{i,v}^k) \neq 0 \\ 1 & \max\limits_{v=1}^{l_i}(Q_{i,v}^k) - \min\limits_{v=1}^{l_i}(Q_{i,v}^k) = 0 \end{cases} \qquad (3)$$

In the above equations, $\max\limits_{v=1}^{l_i}(Q_{i,v}^k)$ is the maximal value of a quality criteria in matrix Q, while $\min\limits_{v=1}^{l_i}(Q_{i,v}^k)$ is the minimal value of a quality criteria in matrix Q. By use these two equations on Q we obtain a matrix $V_{i,j}^k$.

When creating a composite service and, subsequently, when executing it following a user request, the number of component services involved in this composite service may be large, and the number of Web services from which these component services are selected is likely to be even larger. On the other hand, the QoS of the resulting composite service executions is a determinant factor to ensure customer satisfaction, and different users may have different requirements and preferences regarding QoS. For example, a user may require to minimize the execution duration while satisfying certain constraints in terms of price and reputation, while another user may give more importance to the price than to the execution duration. A QoS-aware approach to service composition is therefore needed, which maximizes the QoS of composite service executions by taking into account all the constraints and preferences of the users. So we use the Objectives function to compute the overall quality score for each Web service composition:

$$F = \sum_{i=1}^{m} \left(w_i \times V_i(CWS)\right) \qquad (4)$$

where $0 \le w_i \le 1$, $\sum\limits_{i=1}^{m} w_i = 1$. w_i represents the weight of each QoS criterion and each users can give their preferences on QoS. *CWS* is the service composition with different structures. We will select the solution for service composition based on QoS with the global objective function which has the maximal value of F. Cost objective needs to be maximal in order to participate to optimal solutions. In our model the solutions of our problem must also satisfy some constraints. Only one service in a composition is allocated to each task. In other words, for each task t_i there is a set of Web services $S_i = \{s_{i1}, s_{i2}, \cdots, s_{ij}\}$ that can be assigned to it. However for each task t_i we should only select one web service to execute this task. Given that x_{ij} denotes the selection of Web service s_{ij} to execute the task. t_i. It can be represented by:

$$\sum_{j=1}^{l_i} x_{ij} = 1, x_{ij} \in \{0,1\};$$

where x_{ij} specifies whether nor not a service belongs to a composition and We assume the number of Web services in $S_i = \{s_{i1}, s_{i2}, \cdots, s_{ij}\}$ is l_i. The other constraints concern the user's requirement:

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} Q_{ij}^k \le Q^k \qquad (5)$$

These constraints state that the cost of using the resulting composition should not exceed a given a value Q^k. Two alternative QoS based service selection approaches for composite service execution: one based on local optimization strategy[11] and the other on global planning[12,13]. The local optimization approach performs optimal service selection for each individual task in a composite service without considering QoS constraints spanning multiple tasks and without necessarily leading to optimal overall QoS. By using local strategy we eliminate the service with low QoS from all service sets. Given that there are n service sets $(S_1, S_2, \ldots, S_n)$; each set has l_i Web services and each service has *m* dimension QoS attributes. The input parameters of algorithm1 are QoS threshold matrix QoS_T_{mn} and service sets $(S_1, S_2, \ldots, S_n)$. The output are the new service sets $(S_1^{/}, S_2^{/}, \ldots, S_n^{/})$. The global planning approach on the other hand considers QoS constraints and preferences assigned to a composite service as a whole rather than to individual tasks, and compute optimal plans for composite service executions. In this paper, we proposed the services composition algorithm based on QoS which combined local strategy and global strategy for constraints. Make use of local strategy to eliminate the services with low QoS, and then reduce the problem of services composition plan selection for global QoS guarantee to multi-dimension multi-choice 0-1 knapsack problem which is solved by the heuristic method for improving the efficiency of services composition[10].

We assume that there exits *n* service sets $(S_1, S_2, \ldots, S_n)$; each set has *l* services, and each service has *m* dimension QoS. Figure5 is the experiments result by only run the algorithm proposed and they illuminate the relationship among service composition time, service number in each set is l; the number dimension of OoS is m; and the number of service sets is n.

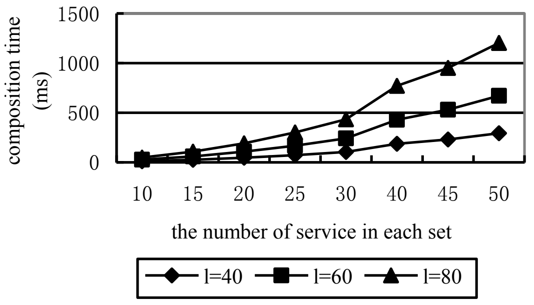

Fig.5 Composition time increases with l and n changing

6. Conclusion

As the use of Web services grows, the problem of service discovery and composition will get more acute. In this paper, the services discovery and composition framework is proposed to solve this problem. Finding appropriate web services according to the ontological domain of the problem was also the aim of our framework. Services discovery algorithm, automated web service composition and QoS optimizing algorithm are used in this framework.

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. Web Services: Concepts, Architecture, and Applications [M]. Springer Verlag (ISBN: 3540440089), June 2003.

[2] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of Web services capabilities. In Proceedings of the First International Semantic Web Conference (ISWC), volume 2342 of Lecture Notes in Computer Science, pages 333–347. Springer-Verlag, 2002.

[3] Sycara, K., Paolucci, M. Ankolekar, A., Srinivasan, N. "Automated Discovery, interaction and composition of Semantic Web Services" , Journal of Web Semantics, Vol 1. no. 1, December 2003, pp. 27-46.

[4] The OWL Services Coalition. OWL-S: Semantic markup for Web Services. Technical report, www.daml.org, 2003.

[5] Wei-hua AI, Zi-lin SONG, Yun-xian HUANG. Web services discovery based on domain ontology [J]. Journal of university of electronic science and technology of China. 2007, 36(3): 506-510.

[6] H.W. Kuhn, The Hungarian method for the assignment problem, Naval ResearchLogistics Quarterly 2 (1955) 83-97.

[7] "New automatic web services composition method based on ontology" Journal of Computational Information System. 2007, 3(4): 1575-1580

[8] OWL-S API. http://www.mindswap.org/2004/owl-s/api/

[9] JENA 2 Ontology API http://jena.sourceforge.net/ontology/index.html

[10] Wei-hua AI, Yun-xian HUANG, Hui ZHANG. Web services composition and optimizing algorithm based on QoS [C]. WiCOM 008. Ocrober 12-17.

[11] Akbar M M, Manning E G, Shoja G C, et al. Heuristic Solutions for the multiple-choice multi-dimension knapsack problem [C]. International Conference on Computational Science. San Francisco, USA. May 2001.

[12] Xiaohui Gu, Klara Nahrstedt, Rong N.Chang, et al. QoS-Assured Service Composition in Managed Service Overlay Networks. IEEE International Conference on Distributed Computing Systems (ICDCS). 2003.

[13] Liangzhao Zeng, Boualem Benatallah, Anne H.H.Ngu, etc. QoS-Aware Middleware for Web Services Composition IEEE Transaction on Software Engineering. 2004, 30(5): 311-327.