Journal of Southeast University (English Edition) Vol. 24, No. 3, pp. 365 – 368 Sept. 2008 ISSN 1003—7985

Service composition algorithm using semantic constraint to implement user personality

Shi Bin Wang Haiyang Cui Lizhen Shi Yuliang

(School of Computer Science and Technology, Shandong University, Jinan 250101, China)

Abstract: In order to improve the efficiency and quality of service composition, a service composition algorithm based on semantic constraint is proposed. First, a user's requirements and services from a service repository are compared with the help of a matching algorithm. The algorithm has two levels and filters out the services which do not match the user's constraint personality requirements. The mechanism can reduce the searching scope at the beginning of the service composition algorithm. Secondly, satisfactions of those selected services for the user's personality requirements are computed and those services, which have the greatest satisfaction value to make up the service composition, are used. The algorithm is evaluated analytically and experimentally based on the efficiency of service composition and satisfaction for the user's personality requirements.

Key words: web service; service composition; semantic similarity; personality; constraint

A web service has the characteristics of platform-independence, self-description, modularization and reuseability which strongly support large-scale web service applications. In finance, tourism and other service areas, more and more services have been published in the form of a web service. As applications of web services expand, inadequacies of web services appear: First, the WSDL description of service is lack of semantic support, which is hard for the computer to understand, so service querying is neither efficient nor accurate; secondly, as the ability of a single web service is limited, the user's complex requirements are difficult to be met.

In order to resolve these problems, we create a related domain ontology and expand the OWL-S description according to the user's personality requirements. OWL-S is a mature semantic web solution. It marks parameters of service with ontology. The ServiceProfile part of OWL-S contains functional descriptions of the service. They are Input, Output, Precondition and Effects, which is called IOPE for short. They are the main parts of semantic matching. The information in ServiceModel is about service composition and data flow. As its capability is poor, we use BPEL as the engine to control the business process in actual applications.

OWL-S supplies semantic descriptions for the service. However, service composition still cannot be achieved. The methods of service composition are mainly in three areas, but all of them have their limitations: the workflow methods need the user's disposal or templet[1-2], so the automation and efficiency is low; the AI methods need to transform service formats and pretreatments which are complicated and difficult to understand and implement[3]; the graph searching methods have significant costs on composition[4-5]. However, the third one is easy to be implemented.

The service composition proposed in this paper focuses on meeting user personality. Suppose that in travelling to Mountain Lu, the scheme for the elderly and the student will be different, because they have their own personalities. The capability of OWL-S's precondition is not enough[6-7], so we add some extended properties, such as weather, price of service, location and so on, to OWL-S while the service is registered.

The new service composition algorithm based on graph searching implements the user's personality requirements. The user's personality requirements are used to filter the services from service repository.

1 Algorithm Description

1.1 Travel ontology

There are two types of ontologies in travel ontology. One is a travel-related ontology, such as flight, ticket, hotel, city and so on; the other is a travel-independent ontology which is the concept used not only in travel but also in daily life, for example time, fare, discount, weather. These two types of ontologies contact each other by the relationship of inherit and citing.

Fig. 1 shows a part of the relationship in a travel ontology. The broken line means inherit and the real line means relationship of citing. As a subclass of Ticket, AirTicket inherits "hasfare" which is the relationship between Ticket and Fare.

1.2 User's personality requirement

Definition 1 User requirement $UR \langle UR_{inp}, UR_{outp}, UR_{per} \rangle$ is a triple. The input of user's requirement UR_{inp} $(In_1, In_2, \ldots, In_k)$ and the output $UR_{outp}(Ou_1, Ou_2, \ldots, Ou_l)$ contain functional parameters of service. $UR_{per} \langle UR_{per}^{con}, UR_{per}^{bet} \rangle$ contains user's personality requirement.

Definition 2 Constraint personality requirement UR_{per}^{con} $(Pe_1, Pe_2, \ldots, Pe_m)$ is the personality requirement which has to be met, just like the location, weather and so on. It can be used to filter redundant services.

Definition 3 Oriented personality requirement UR_{per}^{bet} $(Pe_1, Pe_2, \ldots, Pe_n)$ has fewer constraints than UR_{per}^{con}. For example, the students think more about price, so the train is satisfactory; however, time and comfort is more important to

Received 2008-04-15.

Biographies: Shi Bin(1983—), male, graduate; Wang Haiyang(corresponding author), male, doctor, professor, why@sdu.edu.cn.

Foundation items: The National Natural Science Foundation of China (No. 60673130), the Natural Science Foundation of Shandong Province (No. Y2006G29, Y2007G24, Y2007G38).

Citation: Shi Bin, Wang Haiyang, Cui Lizhen, et al. Service composition algorithm using semantic constraint to implement user personality[J]. Journal of Southeast University (English Edition), 2008, 24(3): 365 – 368.

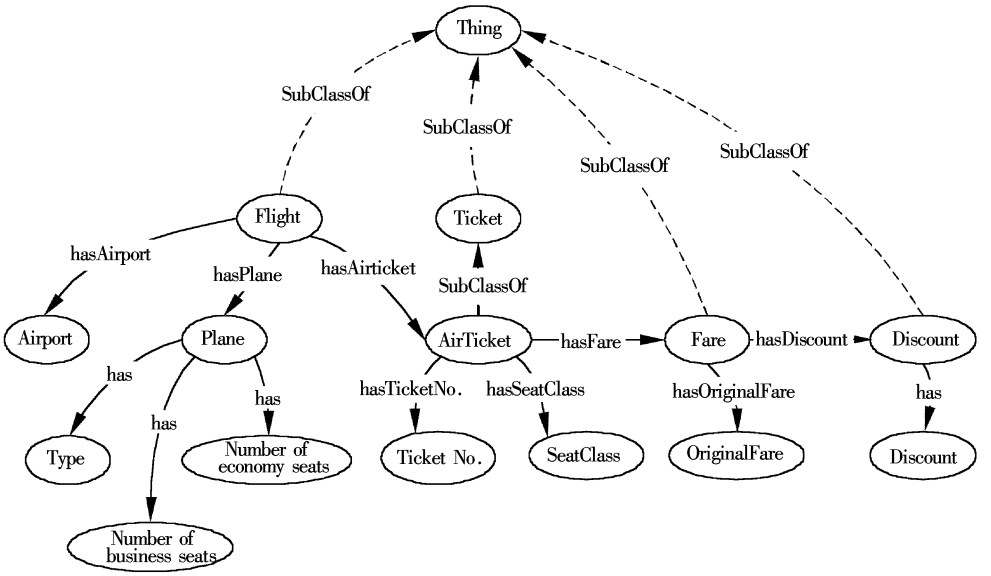

Fig. 1 Part of travel ontology

the merchant, so the plane is preferred. When there is no plane, the train can be used instead. Algorithm 1 shows how to compute satisfaction of service for UR_{per}^{bet}:

Algorithm 1 Service's satisfaction S for UR_{per}^{bet}

Oriented personality requirement $UR_{per}^{bet}(Pe_1, Pe_2, \ldots, Pe_n)$ contains n items that need to be met. For each item Pe_i, we compute the corresponding property's satisfaction S_i. $S_i \in [0, 1]$, the more S_i closes to 1, the better the matching is.

There is a service WS in the Set_{can} and the user's oriented personality requirement $UR_{per}^{bet}(Pe_1, Pe_2, \ldots, Pe_n)$. The WS's satisfaction for UR_{per}^{bet} is

$$S_{WS} = \frac{\sum_{i=1}^{n} \psi_i S_i}{\sum_{i=1}^{n} \psi_i}$$

where ψ_i is the weight of Pe_i in the UR_{per}^{bet}. The enactment of weight should accord with the user's personality. From algorithm 1 we know that S is less than 1 and larger than 0. If all the satisfactions of WS's properties for Pe_i are smaller than the threshold ϕ, WS's satisfaction S_{WS} for UR_{per}^{bet} is 0.

1.3 Service composition

Definition 4 OWL-S description (Ser_C, Ser_{Fun}, Ser_P, Ser_M, Ser_G) is a quintuple vector. Ser_C contains service name, description and other information for users to read. Ser_{Fun} is $\langle In_P, Out_P \rangle$. $In_P = (In_1, In_2, \ldots, In_h)$ is input parameters. $Out_P = (Out_1, Out_2, \ldots, Out_j)$ is output parameters of service. $Ser_P = (P_1, P_2, \ldots, P_q)$ is the extended properties of service. Ser_M and Ser_G are the service information for implement and independent of our discussion. Some extended properties of service are shown as follows:

```
〈personality〉
  〈personality: addproperties〉
    〈personality: addproperty〉
      〈personality: parameterType rdf: datatype = "travel-ontology#city"/〉
      〈personality: dataproperty rdf: dataname = "name"/〉
      〈personality: value rdf: value = "Guilin"/〉
    〈/personality: addproperty〉
    〈personality: addproperty〉
      〈personality: parameterType rdf: datatype = "travel-ontology#fare"/〉
      〈personality: dataproperty rdf: dataname = "actualFare"/〉
      〈personality: dataconstraint rdf: constraint = "no-less-than"/〉
      〈personality: value rdf: value = "500"/〉
    〈/personality: addproperty〉
    〈personality: addproperty〉
      〈personality: parameterType rdf: datatype = "travel-ontology#hotel"/〉
      〈personality: dataproperty rdf: dataname = "star"/〉
      〈personality: value rdf: value = "3"/〉
    〈/personality: addproperty〉
  〈/personality: addproperties〉
〈/personality〉
```

From the above extended properties we know the location of service is Guilin, the lowest cost is RMB 500 yuan and the hotel is a three-star hotel.

Definition 5 The semantic matching in this paper has two levels. In the lower level, the matching can be divided into four types: Exact, Plugin, Subsume and Fail. The input of user requirement UR_i is In. The output is Ou. The item in service input In_P is In_s. The item in service output Out_P is Out_s. If the following three conditions can be met, the matching on the lower level is successful:

1) Exact matching: $In \Leftrightarrow In_s$ or $Ou \Leftrightarrow Out_s$;

2) Plugin matching: In_s contains In;

3) Subsume matching: Ou contains Out_s

All the other conditions are failing. The upper level is logical-based. Users' input is UR_{inp}. What the users want is UR_{outp}. Service input is In_P. Service output is Out_P. If the following conditions can be met, the service matches the user's

personality requirements:

1) $UR_{inp} \Leftrightarrow In_P$ or $UR_{outp} \Leftrightarrow Out_P$;

2) $\forall In(In \in In_P \Rightarrow In \in UR_{inp})$, viz. $UR_{inp} \supseteq In_P$;

3) $\forall Ou(Ou \in UR_{outp} \Rightarrow Ou \in Out_P)$, viz. $UR_{outp} \subseteq Out_P$.

Algorithm 2 The service composition algorithm based on user personality

Input: User requirement $UR(UR_{inp}, UR_{outp}, UR_{per})$, service repository $R(WS)$;

Output: Service composition.

Step 1 Search in repository R. If the output of service WS and UR_{outp} matches, go to step 3. Use UR_{per}^{con} to filter services from repository R and add the matching service to the candidate service set Set_{can};

Step 2 Search in Set_{can}. If the input of service WS matches user requirement UR_{inp}, add WS into selected service set Set_{cur} and delete it from Set_{can}. Then merge the output of WS into the set $OutP_{cur}$. If $OutP_{cur}$ matches UR_{outp} after searching Set_{can}, go to step 3; else if Set_{can} is not null, merge $OutP_{cur}$ into UR_{inp} and go to step 2; if Set_{can} is null or the services number in Set_{can} has no change, go to step 4;

Step 3 Compute the satisfactions of services with the same input and output parameters in Set_{cur}. Then select the biggest one as a component of service composition and add it to the component list WS_{List}. Go to step 5;

Step 4 The composition fails;

Step 5 Arrange the services in WS_{List} in reverse. The result is the service composition.

Exposition of algorithm 2: The aim of filtering the services and comparing the services number of Set_{can} in step 2 is to avoid ending up in cycles. Our method gains a preferable result, and the costs of time and space are little. From algorithm 2, we know that the algorithm just contacts service repository R once and the rest of the computing occurs in Set_{cur}, which can reduce the costs of time greatly.

2 Experiment and Analysis

In order to validate the efficiency of our method, we choose nine groups of services. The scale of services is between 100 and 600.

From Fig. 2, we can know that as the number of services increases, the effects of constraint personality requirements is more and more obvious. The reason is that irrelevant services need not to be added to Set_{can} and need not be involved in the following computation. So the more complicated the composition is, the more time can be reduced.

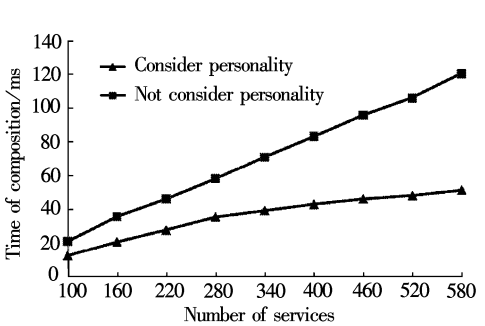

Fig. 2 Time cost of service composition

Fig. 3 reveals user satisfaction for service composition. From Fig. 3, we can know that when the personality requirement is considered, user satisfaction is enhanced as the number of services increases.

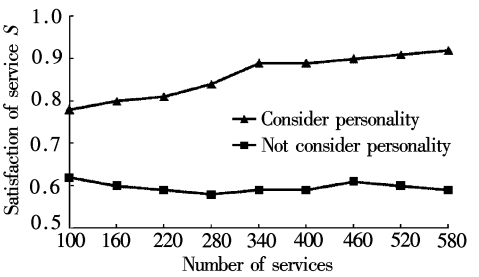

Fig. 3 Satisfaction of service composition

However, when we do not consider the personality requirement, the improvement of satisfaction is not obvious. So we can conclude that the satisfaction cannot recede as the number of services increases when the personality requirement is considered.

3 Conclusion

With the constraint of the user's requirements, the service composition algorithm deletes many redundant services at the beginning of service composition. It reduces the searching space of services, accelerates the service composition and provides a service composition which satisfies users most according to their requirements. Proved by experiment, this algorithm not only improves speed of composition but also improves user satisfaction for composition results. In future, we will consider the relationship between services in order to provide a more intelligent service composition.

References

[1] Aversano L, Canfora G, Ciampi A. An algorithm for web service discovery through their composition [C]//*Proc of the* 2004 *IEEE International Conference on Web Services*. San Diego, California, USA, 2004: 332 – 341.

[2] Hu H T, Li G, Han Y B. An approach to business-user-oriented large-granularity service composition [J]. *Chinese Journal of Computers*, 2005, **28**(4): 694 – 703. (in Chinese)

[3] Xu M, Chen J L. Integrating semantic business policy into web service composition [C]//*Proc of the* 1*st International Workshop on Service Oriented Modeling*. Berlin: GITO-Verlag, 2006: 84 – 95.

[4] Xie X Q, Chen K Y, Li J Z. A composition oriented and graph-based service search method [C]//*Proc of the* 1*st Asian Semantic Web Conference*. Beijing, China, 2006: 530 – 536.

[5] Hashemian S V, Mavaddat F. A graph-based framework for composition of stateless web services[C]//*Proc of the European Conference on Web Services*. Zürich, 2006: 75 – 86.

[6] Xu M, Chen J L, Peng Y, et al. Service relationship ontology-based web service creation [J]. *Journal of Software*, 2008, **19**(3): 545 – 556. (in Chinese)

[7] Liu Z Z, Wang H M, Zhou B. A two layered P2P model for semantic discovery [J]. *Journal of Software*, 2007, **18**(8): 1922 – 1932. (in Chinese)

利用语义约束实现用户个性化的服务组合算法

史 斌 王海洋 崔立真 史玉良

(山东大学计算机科学与技术学院,济南 250101)

摘要:为了提高服务组合的质量和效率,提出了一种基于语义约束的服务组合算法. 算法中,首先利用 2 层语义匹配算法对用户需求和服务库中的服务进行比较,利用用户约束性个性化需求对符合匹配的服务进行筛选,从而在第一时间缩小了服务组合算法执行过程中服务的搜索空间,然后对候选的服务组合组件进行用户趋向性个性化需求计算,选择其中最符合用户个性化需求的服务构成最终的服务组合. 通过仿真实验证明,该算法有效提高了服务组合算法的运行效率,并且保证了用户的个性化需求得到最大满足.

关键词:web 服务;服务组合;语义相似度;个性化;约束

中图分类号:TP311