Journal of Software Engineering

ISSN 1819-4311

Journal of Software Engineering 7 (2): 68-76, 2013
ISSN 1819-4311 / DOI: 10.3923/jse.2013.68.76

A Web Service Discovery and Composition Method Based on Service Classes

Wei Liu, Yuyue Du and Chun Yan
College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

Corresponding Author: Wei Liu, College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

ABSTRACT

A service class can be formed by Web services with the same function from an abstract operation. Some operations for the service class are presented to reflect its self-adaptation. On the basis of the service class, Web service discovery and composition algorithms are given. In Web service discovery process, a projective operation is proposed and a selection algorithm is provided to choose the optimal Web services and in Web service composition process, some service composition operations are given. The Web service discovery and composition efficiency is improved based on service classes. Finally, the effectiveness of the proposed methods is illustrated by the service composition for travel planning.

Key words: Web service, service class, service discovery, service composition, service composition operation

INTRODUCTION

With the development of Web services, the number of Web services is increasing dramatically and more pressure is encountered in Web service discovery and composition. Many services with the same functions are published by different service providers. This will result in the redundancy of some services with the same functions. How to alleviate the burden of network and eliminate the redundancy is becoming a hard task. Thus, the research on Web services discovery is becoming a hot topic (Huhns and Singh, 2005). And on the other hand, it is difficult to meet the needs of users only by a single Web service. So Web service composition is widely investigated.

At present, many standards of Web service composition have been proposed (Curbera *et al.*, 2002) such as the Simple Object Access Protocol (SOAP) used to exchange the messages among services, service providers and clients, Web Service Description Language (WSDL) used to describe service functions and interface and Universal Description, Discovery and Integration (UDDI) used to advertise and discover services. But some complex combinations of Web services cannot be realized based on the standards. Many approaches and technologies have been applied to the composition and verification of Web services, such as Agent, workflows, Petri nets, semantic web and Pi calculus. A workflow-based Web service composition method was introduced (Cardoso and Sheth, 2003). A model was proposed based on Petri nets and situation calculus and the simulation, verification, composition of Web services were made (Li *et al.*, 2011). Business Process Execution Language for Web Services (BPEL4WS) was transformed into a service-oriented Petri net and Web service composition was verified. Colored Petri Nets were applied to describe Web service composition (Azgomi and Entezari-Maleki, 2010). The concept of service community (Benatallah *et al.*, 2003; Maamar *et al.*, 2010), meta service (Li *et al.*, 2007) and service cluster (Qi *et al.*, 2012) are defined to facilitate service composition.

Different from the previous researches on Web service composition, a web service composition method based on service class is presented in this study. The architecture of service composition presented in the study is shown in Fig. 1. There mainly are two kinds of users: requestor and provider. The providers can register some services to the service registry and then some resolving work can be done based on the domain ontology. According to the service function, the atomic services can be classified as different classes through an abstract operation and a service class includes the services with the same function. Thus, the number of the service classes is much smaller than that of the atomic services and the discovery efficiency can be improved. As is well known, there are some dynamic changes for the service classes, thus an updating strategy is proposed. For example, some new services need to be added to service classes when providers provide them (Addition operation⊕) some services existing in the service classes need to be removed when they are unavailable or out of date (Subtraction operation©) and some corresponding changes should be made for service classes when the service provider change the service information (Modification operation®). The service composition can be implemented based on the service classes by the composition engine and the corresponding atomic services achieving the function of service classes can be obtained through projection operations, then service execution can be realized. Finally, the results can be returned to the requestor.

The rest of the study is organized as follows. Basic concepts for Web services and service class are presented and some operations for the service class are proposed in Section 2. Section 3 is devoted to Web service discovery and composition process: a projective operation method is proposed, a selection strategy is provided to choose the optimal Web services and some composition operations are given. The effectiveness of the proposed methods is illustrated by an example.

The Web service discovery and composition method is given based on service classes in the study. The Web service discovery and composition efficiency is improved based on service classes.

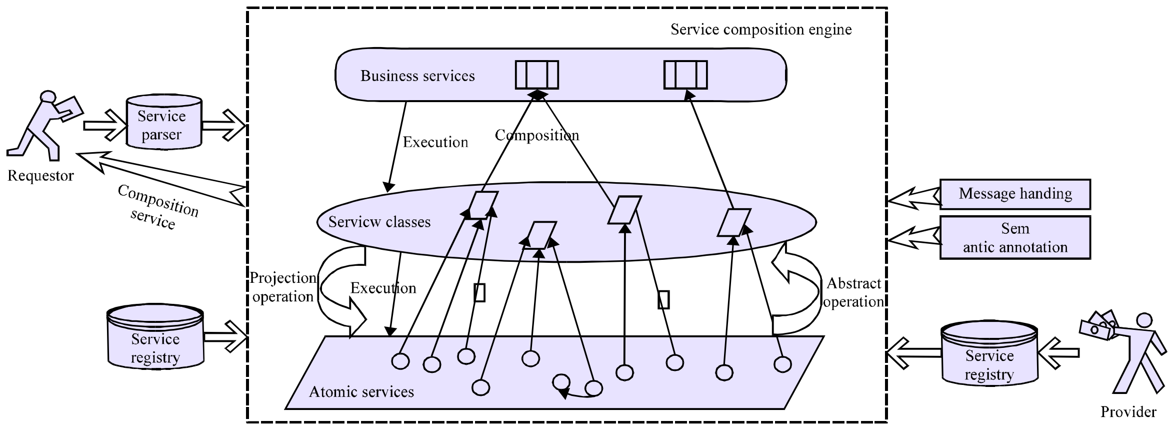

Fig. 1: The architecture of service composition based on service classes

WEB SERVICES AND SERVICE CLASS

Some notations are given first. Let $S = \{s_1, s_2, ..., s_n\}$ be a finite set. $|S|$ represents the number of elements in S. Sets U and V are disjoint if $U \cap V = \varnothing$. If $\forall s \in U \Rightarrow s \in V$ and $\forall r \in V \Rightarrow r \in U$, then $U = V$. $U \setminus V = \{p \mid p \in U \text{ and } p \notin V\}$. $\mathbb{N}$ is a natural number set, i.e., $\mathbb{N} = \{0, 1, 2, ...\}$; $\mathbb{N}_k = \{0, 1, 2, ..., k\}$; $\mathbb{N}^+ = \mathbb{N} \setminus 0\}$ and $\mathbb{N}_k^+ = \mathbb{N}_k \setminus \{0\}$.

In order to find the suitable Web services quickly and effectively, Web services with the same function are combined into a new service class. First an atomic service is defined as a simple operation receiving a group of input parameters and offering a group of output parameters. the Semantic Web version is followed and the service descriptions are enriched by annotating their parameters with semantic concepts taken from domain ontology. Then services with the same functions can be classified as a service class. Some operations for the service class are also presented.

Basic concepts

Definition 1: Atomic service: An atomic service is expressed as a 6-tuple s = (Na, SF, I, O, QoS, URL), where, Na denotes the service name, SF is a concept taken from domain ontology and denotes the service function, I is a set of input parameters from domain ontology, O is a set of output parameters from domain ontology, QoS denotes the service quality and it can be calculated by the properties such as usability, response time, cost and reliability and URL is short for Uniform Resource Locator denoting the service address and it is unique.

In this study, an atomic service is defined as a simple operation with a name and a function from domain ontology and receives a group of input parameters and offers a group of output parameters. Besides, a QoS (Quality of Services) and an URL (Uniform Resource Locator) are attached.

Definition 2: Service request: A service request given by a service requestor can be expressed as a 4-tuple $r = (SF_r, I_r, O_r, QoS_r)$, where, SF_r is a concept taken from domain ontology and denotes the service function needed by the requestor, I_r is a set of known information, i.e., a list of input parameters from domain ontology provided by the requestor, O_r denotes a set of needs, i.e., a list of output parameters from domain ontology requested by the requestor and QoS_r denotes a set of constrains on service quality made by the requestor.

Notice that a requestor needs a list of outputs while offering a list of inputs and a set of constrains on service quality.

For example, $s1 = (Na_1, SF_1, I_1, O_1, QoS_1, URL_1)$ is a train ticket query atomic service, where, SF_1= TrainTicketQuery, I_1 = {Departure, Arrival, Date} and O_1 = {Time, Price, Type} with the concepts from travel ontology. And there are two other train ticket query atomic services, such as $s_2 = (Na_2, SF_2, I_2, O_2, QoS_2, URL_2)$ and $s_3 = (Na_3, SF_3, I_3, O_3, QoS_3, URL_3)$, $SF_2 = SF_3$ = TrainTicketQuery, I_2 = {Departure, Arrival}, O_2 = {Type, Time, Price}, I_3 = {Departure, Arrival, Date} and O_3 = {Price, Time, Train seatings}. Since $SF_1 = SF_2 = SF_3$, the services s_1, s_2, s_3 with the same function can be classified as a service class.

Definition 3: Service class: Let $s_1, s_2, ..., s_n$ be n atomic services and $\forall i, j \in \mathbb{N}_n^+$, $SF_i = SF_j$. A service class is expressed as a 3-tuple sc = (Na, AS, SCF) where, Na denotes the name of the service class, $AS = \{s_1, s_2, ..., s_n\}$ is a set of atomic services and $SCF = SF_i$, $i \in \mathbb{N}_n^+$.

Notice that a service class includes several services with the same function though there may be great differences between the other variables of these services. And in the definition, SCF denotes service class function and it can be described by the function of any atomic services in sc, because all the services in sc can complete the same function. Thus, the number of service classes is much smaller than that of the atomic services in network and the atomic services can be managed efficiently.

In the above example, there exist three services s_1, s_2 and s_3 of querying train tickets, where $SF_1 = SF_2 = SF_3$ = TrainTicketQuery. The three atomic services can be classified as a service class, denoted as sc = (Na, AS, SCF) where, AS = $\{s_1, s_2, s_3\}$, SCF = TrainTicketQuery.

Operations for the service class: In order to manage atomic services in a service class efficiently, some operations are proposed in this subsection. This reflects the self-adaptation of the service class.

Definition 4: Abstract operation: Let AS = $\{s_1, s_2,..., s_n\}$ be a set of atomic services, where, $s_i = (Na_i, SF_i, I_i, O_i, QoS_i, URL_i)$ be an atomic service, $i \in N_n^+$. P(AS) = $\{sc_1, sc_2,..., sc_m\}$ is the abstract operation on AS, where, $sc_j = (Na_j, SF_j, S_j)$ is a service class.

Notice that through an abstract operation, the atomic services can form a service class set. And this service class set satisfies some properties.

Theorem 1: Suppose AS = $\{s_1, s_2,..., s_n\}$ be a set of an atomic services, P(AS) = $\{sc_1, sc_2,..., sc_m\}$ is a partition of AS.

Proof: According to the definition of the partition, the following conditions should be satisfied.

- For $\forall s_i \in AS$, $s_i \in sc_j$ and $sc_j \in P(AS)$ and
- If $sc_i \neq sc_j$, then $sc_i \cap sc_j = \varnothing$

According to definition 3 and 4, if $Sf_i = SCF_j$ and $sc_j \notin P(AS)$, then $P(AS) = P(AS) \cup \{sc_j\}$; if $Sf_i = SCF_j$ and $sc_j \in P(AS)$, then $sc_j = sc_j \cup \{s_i\}$. Thus, for $\forall s_i \in AS$, $s_i \in sc_j$ and $sc_j \in P(AS)$.

Suppose sc_j, $sc_k \in P(AS)$, $sc_i \cap sc_j \neq \varnothing$ and $s_i \in sc_j \cap sc_k$, then $s_i \in sc_j$ and $s_i \in sc_k$. Since service classes are constructed according to the service function and on the basis of definition 3, $Sf_i = SCF_j$ and $Sf_i = SCF_k$, thus $SCF_j = SCF_k$, i.e., $sc_i = sc_j$.

Therefore, the conclusion is correct.

After the abstract operation on AS is finished, a service class set P(AS) is constructed. But in fact services in the network are always changed. The service class should have the property of self-adaptation. Thus some operations are proposed in this subsection to manage atomic services in a service class reasonably and efficiently.

Definition 5: Addition operation: Let sc = (Na, S, SCF) be a service class, where, S = $\{s_1, s_2,..., s_n\}$. $S = S \cup \{s'\}$ is the new service set in sc, denoted by $sc = sc \oplus s'$, if there exists a new service $s' \notin S$, $s' = (Na', SF', I', O', QoS', URL')$ and $SCF = SF'$.

Notice that the Addition operation means that some new services need to be added to service classes when the providers provide them.

Definition 6: Subtraction operation: Let sc = (Na, S, SCF) be a service class, where S = $\{s_1, s_2,..., s_n\}$. $\forall s \in S$, s = (Na, SF, I, O, QoS, URL), sc = sc©s denotes deleting s from the set S in sc, i.e., $S = S \setminus \{s\}$.

Notice that the subtraction operation occurs when some services existing in the service class need to be removed when they are unavailable or out of date.

Definition 7: Modification operation: Let sc = (Na, S, SCF) be a service class, where, $S = \{s_1, s_2,..., s_n\}$. For s∈S, s = (Na, SF, I, O, QoS, URL), sc = sc®s denotes making some changes to s from the set S in sc, if no less than one of variables Na, SF, I, O, QoS and URL are changed except for the function SF.

When the modification operation happens, the previous variables of the corresponding service are updated by the new values provided by the provider. Some corresponding changes should be made for the services existing in the service class when the service providers change the service information.

There are some dynamic changes for the service classes and based on the updating strategy in the above definitions, an algorithm for updating the service class is proposed in the following:

Algorithm 1: Algorithm for updating a service class

```
    Inpnt: P(AS) = {$sc_1$, $sc_2$,... $sc_m$} is the abstract operation on an atomic service set AS, sc = (Na, S, SCF)∈P(AS), op∈{⊕, ©, ®}, s = (Na,
SF, I, O, QoS, URL)
    Outpnt: The updated P(AS)
    if (s∉AS)
    {if (SCF = SF)
    {sc = sc⊕s;}
    else if (∃sc′∈P(AS), SCF′ = SF)
    {sc′ = sc′⊕s;}
else
    {SCF′′ = SF; sc′′ = (Na′′, {s}, SCF′′);
P(AS) = P(AS)∪{sc′′};} }
else if (s∈S is not obtained)
    {sc = sc©s;
    if (S = ∅)
    {P(AS) = P(AS)\sc;}}
    else if (s∈S is changed)
    {if (SCF = SF)
    {sc = sc®s;}
    else
    {sc = sc©s;
    if (S = ∅)
    {P(AS) = P(AS)\sc;}
    if(∃sc′∈P(AS), SCF′ = SF)
    {sc′ = sc′⊕s;}
else
    {sc′′ = (Na′′, {s}, SCF′′); P(AS) = P(AS)∪{sc′′};}}
Return P(AS);
```

SERVICE DISCOVERY AND COMPOSITION

Web service discovery and composition are the study focus in the industrial and academic circles and many methods have been proposed. In this study, a composition method based on service class is proposed.

Operations for service discovery: The semantic web version is followed and the service descriptions are enriched by annotating their parameters with semantic concepts taken from domain ontology (Paliwal *et al.*, 2011; Klusch *et al.*, 2009). Let c_1, c_2 be concepts from domain ontology and the similarity between c_1 and c_2 is defined as follows:

$$\mathrm{sim}(c_1, c_2) = \frac{\mathrm{dis}_{\max} - \mathrm{dis}(c_1, c_2)}{\mathrm{dis}_{\max} - \mathrm{dis}_{\min}}$$

where, $\mathrm{dis}(c_1, c_2)$ is the distance between c_1 and c_2, i.e., the edge-number of the shortest path between the two concepts in the concept tree. $\mathrm{dis}_{\max}$ and $\mathrm{dis}_{\min}$, respectively are the maximum and minimum values of $\mathrm{dis}(c_i, c_j)$, where, $c_i, c_j \in C$ and $i, j \in N_{|C|}^{+}$.

Definition 8: Projective operation: Let $P(AS) = \{sc_1, sc_2,..., sc_m\}$ be the abstract operation on an atomic service set AS and $r = (SF_r, I_r, O_r, QoS_r)$ be a service request. There are three levels of projective operation defined as follows:

- $f_{L1}(P(AS)) = \{sc \mid sc \in P(AS) \text{ and } L1|_{sc} = \bullet T\bullet\}$ is a projective operation on P(AS), where sc = (Na, S, SCF), L1: $\mathrm{sim}(SCF, SF_r) \geq \sigma$ is a proposition, $\sigma > 0$ is a real number and $L1|_{sc} = \bullet T\bullet$ denotes that the logical value of L1 is true for the function SCF in sc.
- Let $SC = f_{L1}(P(AS))$ and $sc = (Na, S, SCF) \in SC$ be a service class, where, $S = \{s_1, s_2,..., s_n\}$ and $s = (Na, SF, I, O, QoS, URL) \in S$. $f_{L2}(sc) = \{s \mid s \in S \text{ and } L2|_s = \bullet T\bullet\}$ is a projective operation on sc, where $L2 = l(I) \wedge l(O)$, $l(I)$: $I \subseteq I_r$ and $l(O)$: $O \supseteq O_r$ are propositions and $L2|_s = \bullet T\bullet$ denotes that the logical value of L2 is true for the variable values I and O in s
- Let $f_{L1}(P(AS)) = \{sc_1, sc_2,..., sc_k\}$ and $SS = f_{L2}(sc_1) \cup f_{L2}(sc_2) \cup ... \cup f_{L2}(sc_k)$. $f_{L3}(SS) = \{s \mid s \in SS \text{ and } L3|_s = \bullet T\bullet\}$ is a projective operation on SS, where L3: $QoS \geq QoS_r$ is a proposition

Notice that, given a certain service request, $f_{L1}(P(AS))$ provides us a set of service classes meeting the request; then $f_{L2}(sc)$ returns a set of atomic services in *sc* according to the input and output parameters; and at last, $f_{L3}(SS)$ shows a set of services that satisfy the quality of service request. Let $CS = f_{L3}(SS)$ be set of candidate services that may satisfy a service request. Then the optimal service should be selected from CS and the following choosing strategy is used.

Definition 9: Matching value: Let $r = (SF_r, I_r, O_r, QoS_r)$ be a service request and $s = (Na, SF, I, O, QoS, URL) \in CS$. M(r, s) is a matching value of r and s, where, $M(r,s) = \omega_1\, \mathrm{sim}(SF, SF_r) + \omega_2/(|I_r - I| + \theta) + \omega_3/(|O - O_r| + \theta) + \omega_4(QoS - QoS_r)$, $\theta > 0$, ω_1, ω_2, ω_3 and $\omega_4 \in [0,1]$ are the weights and $\sum_{i=1}^{4}(\omega_i) = 1$.

Notice that, given a service request r, each service s in candidate service set from definition 8 has a matching value according to definition 9. $\mathrm{sim}(SF, SF_r)$ denote the similarity between the functions of r and s; $\omega_2/(|I_r - I| + \theta)$ denotes the redundant degree of the input parameters of r and s, where $|I_r - I|$ denotes the number of the disjoint input parameters and if there are no redundant parameters, i.e., $I_r = I$, then $|I_r - I| = 0$ and $\omega_2/(|I_r - I| + \theta)$ reaches maximum value; similarly, $\omega_3/(|O - O_r| + \theta)$ denotes the redundant degree of the output parameters of r and s and $QoS - QoS_r$ denotes the quality value of s better than r.

Algorithm for service discovery: According to definition 8 and 9, the service discovery algorithm can be given as follows:

Algorithm 2: Algorithm for service discovery

```
Inpnt: A set of service classes $P(AS) = \{sc_1, sc_2,..., sc_m\}$ and a service request $r = (SF_r, I_r, O_r, QoS_r)$
Outpnt: A sorted set of service that satisfy the service request
compute $f_{L1}(P(AS))$ by (1) in definition 8 and let $SC = f_{L1}(P(AS)) = \{sc_1, sc_2,..., sc_k\}$;
for each $sc_i \in SC$ do
  {compute $f_{L2}(sc_i)$ by (2) in definition 8;}
$SS = f_{L2}(sc_1) \cup f_{L2}(sc_2) \cup ... \cup f_{L2}(sc_k)$;
compute $f_{L3}(SS)$ by (3) in Definition 8 and let $CS = f_{L3}(SS) = \{s_1, s_2,..., s_n\}$;
for each $s_j \in SC$ do
  {compute $M(r, s_j)$ by definition 9;}
$s_1, s_2,...$ and $s_n$ are sorted according to $M(r, s_j)$ and the sorted CS are $s_1', s_2',..., s_n'$, where $M(r, s_k') = M(r, s_l')$, i.e., $s_k'$ is better than $s_l'$, if $\forall k, l \in N_n^+$ and $k<l$;
Return the sorted CS: $s_1', s_2',..., s_n'$;
Given a set of service classes $P(AS) = \{sc_1, sc_2,..., sc_m\}$ and a service request $r = (SF_r, I_r, O_r, QoS_r)$, a sorted atomic services CS: $s_1', s_2',..., s_n'$ is obtained from algorithm 2 and they all satisfy r and the prior services in CS are better than the posterior ones. $s_1'$ should be chosen first and if $s_1'$ cannot be obtained, then $s_1'$ is substituted by $s_2'$. Similarly, atomic services can be selected in order until an available service is obtained.
```

Operations for service composition: From the composition patterns of workflow nets (Qi *et al.*, 2012), there are mainly four basic patterns in the composition of service class, i.e., sequence, parallelism, alternative and iteration. Now these patterns are proposed briefly.

Let $P(AS) = \{sc_1, sc_2,..., sc_m\}$, where sc_i, sc_j are service classes, $i, j \in N_m^+$. The operations on service classes can be defined as follows:

- Sequence operation $\bullet$: $sc_i \bullet sc_j$ represents a sequence completion from sc_i to sc_j
- Alternative operation $\Diamond$: $sc_i \Diamond sc_j$ represents an alternative completion of sc_i and sc_j and precisely one of them is executed
- Parallel operation $||$: $sc_i || sc_j$ represents a parallel completion of sc_i and sc_j and they are executed concurrently
- Iteration operation μ_n: $\mu_n sc_i$ represents an iteration completion of sc_i and it performs n times repeatedly, where $n \in N^+$

The Web service composition can be illustrated by an example in the following:

SERVICE COMPOSITIONS FOR TRAVEL PLANNING

Service composition can be completed based on the service classes by the composition engine and the corresponding atomic services achieving the service classes can be obtained through projection operations, then service execution can be realized. And an example is given in the following to illustrate the proposed method in this study.

To make a travel plan, some service classes should be invoked and composed to satisfy the request. In general, for example, eight service classes are needed: Weather Forecast, Place Choose, TrainTicketQuery, TrainTicketPay, AirTicketQuery, AirTicketPay, HotelQuery and HotelPay. The control structure on these service classes is shown in Fig. 2 and the service classes are denoted by squares, respectively.

There are n atomic services $si = (Na_i, SF_i, I_i, O_i, QoS_i, URL_i)$, $i \in Nn^+$ and they are classified as several service classes by the attraction operation. Here the service classes with the function of querying and paying for train tickets are taken as examples, respectively.

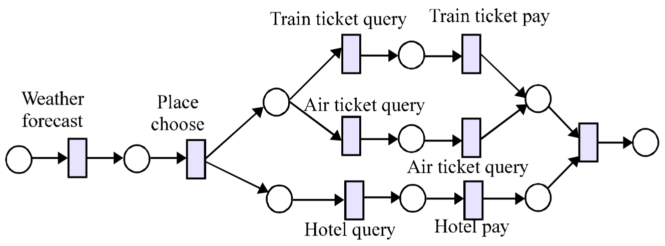

Fig. 2: The control structure on service classes in travel planning

Let $sc_1 = (Na_1, S_1, SCF_1)$ where $S_1 = \{s_1, s_2, s_3\}$, SCF_1 = TrainTicketQuery, $sc_2 = (Na_2, S_2, SCF_2)$ where, $S_2 = \{s_4, s_5\}$, SCF_2 = TrainTicketPay. The details of $s_i = (Na_i, SF_i, I_i, O_i, QoS_i, URL_i)$, $i \in N_5^+$ are as follows:

- s_1 = (Na_1, TrainTicketQuery, {Departure, Arrival, Date}, {Time, Price, Type}, QoS_1, URL_1)
- s_2 = (Na_2, TrainTicketQuery, {Departure, Arrival}, {Time, Price, Type}, QoS_2, URL_2)
- s_3 = (Na_3, TrainTicketQuery, {Departure, Arrival, Date}, {Time, Price, Train seatings}, QoS_3, URL_3)
- s_4 = (Na_4, TrainTicketPay, {AccountInf, Price}, {DealFail, DealSucceed}, QoS_4, URL_4)
- s_5 = (Na_5, TrainTicketPay, {AccountInf, Price}, {DealFail, DealSucceed}, QoS_5, URL_5)

According to Algorithm 1, if a provider provides a new service s_6 = (Na_6, TrainTicketQuery, {Departure, Arrival, Date}, {Time, Price, Type}, QoS_6, URL_6), then the function of s_6 TrainTicketQuery which is the same as sc_1, thus $sc_1 = sc_1 \oplus s_6$; if service s_3 is invalid, then $sc_1 = sc_1 © s_3$; and if s_1 is changed that $I_1 = I_1 \cup \{price\}$, then $sc = sc ® s_1$.

According to the definitions of the operations between service classes, the sequence composition of two service classes sc_1 and sc_2 can be denoted as $sc_1 \bullet sc_2$. The sequence composition can also be denoted with the function of the service classes, i.e., TrainTticketQuery•TrainTicketPay.

Thus, the process of the travel planning service can be described by the operations between service classes, i.e., Weather Forecast•Place Choose•((TrainTicketQuery•TrainTicketPay) ◊(TrainTicketQuery•TrainTicketPay) | | (HotelQuery•HotelPay)).

At last, according to Algorithm 2, a set of sorted services for each service class in the above process can be obtained and then available atomic services can be found. Services can be chosen from the sorted service sets in order and then the process of the travel planning service composition with atomic services is constructed.

CONCLUSIONS

A service class can be formed by Web services with the same functions. In the paper, the operations related to services classes such as the abstract operation, the addition operation, the subtraction operation, the modification operation, projection operations are presented. And on the basis of the service class, the service discovery algorithm and the operations for service composition are given. Future work will investigate automatic service composition based on service classes.

ACKNOWLEDGMENTS

This study is supported by the National Basic Research Program of China under grant 2010CB328101, The National Natural Science Foundation of China under grants 61170078 and

71001057, the Doctoral Program of Higher Education of the specialized research fund of China under grant 20113718110004, The Scientific and Technological Developing Program of Shandong Province of China under grant 2011GGX10114, The Project of Shandong Province Higher Educational Science and Technology Program under Grant number J12LN11, the China's Post-doctoral Science Fund under grant 2012M521362 and the SDUST Research Fund of China under grants 2011KYTD102 and 2011KYTD104.

REFERENCES

Azgomi, M.A. and R. Entezari-Maleki, 2010. Task scheduling modelling and reliability evaluation of grid services using coloured petri nets. Fut. Gen. Comp. Sys., 26: 1141-1150.

Benatallah, B., Q.Z. Sheng, M. Dumas, 2003. The self-serv environment for Web services composition. IEEE Internet Comput., 7: 40-48.

Cardoso, J. and A. Sheth, 2003. Semantic E-workflow composition. J. Intell. Inform. Sys., 21: 191-225.

Curbera, F., M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, 2002. Unraveling the web services web: An introduction to SOAP, WSDL and UDDI. IEEE Internet Comput., 6: 86-93.

Huhns, M.N. and M.P. Singh, 2005. Service-oriented computing: Key concepts and principles. IEEE J. Internet Comput., 9: 75-81.

Klusch, M., G. Fries and K. Sycara, 2009. OWLS-MX: A hybrid semantic web service matchmaker for owl-s services. J. Web Sem., 7: 121-133.

Li, H., H. Wang and L. Cui, 2007. Automatic composition of web services based on rules and meta-services. Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design, April 26-28, 2007, Melbourne, Vic., pp: 496-501.

Li, X., Y. Fan, Q.Z. Sheng, Z. Maamar and H. Zhu, 2011. A petri net approach to analyzing behavioral compatibility and similarity of web services. IEEE Trans. Syst. Man Cybern., 4: 510-521.

Maamar, Z., L.K. Wives, G. Al-Khatib, Q. Z.Sheng, A.R.D. de Vit and D. Benslimane, 2010. From communities of web services to marts of composite web serivces. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and applications, April 20-23, 2010, Perth, WA, pp: 958-965.

Paliwal, A.V., B. Shafiq, J. Vaidya, H. Xiong and N. Adam, 2012.. Semantics based automated service discovery. IEEE Tran. Ser. Comp., 5: 260-260.

Qi, L., Y.Y. Du and W.J. Luan, 2012. Cluster-oriented service composition modeling method based on logical net units. Proceedings of the 9th IEEE International Conference on Networking Sensing and Control, April 11-14, 2012, Beijing, pp: 112-117.