The Petri Net model of the requirements and service composition

Chen Zhijuan
College of Mathematics, Physics and Information Engineering
Zhe Jiang Normal University
Jinhua, China
czj19871020@126.com

Ye Ronghua
Institute of Computer Science and Technology
Zhe Jiang Normal University
Jinhua, China
rhye@zjnu.edu.cn

***Abstract*—In the service-oriented environment, the composition of multiple web services is always used to satisfy the given request. To meet the request, the compositions of the services are various. Aim at such difficulty, described with the environment ontology, the relative theory of Petri net is proposed to build up the service composition requirement model. After simplify the model, an algorithm of building a reachability tree is proposed. Then all the possible service compositions are got through an algorithm similar as depth-first search. At last, we use the classic example of travel arrangement to verify the theory above all.**

Keywords- Environment Ontology; Petri Net; reachable tree; service composition

I. INTRODUCTION

SOC and SOA have been the hot issues of software area recently, which embodies the software reuse, and structures the develop model of application in the way of service and its composition[1]. Web service is a module application component which can be published, discovered, and located. As result, how to describe the service becomes one of the most important issues of SOC, and the link between requirements of users and services, as well as the base of service discovery and service composition. Now it has been used widely, but it is a challenge to implement SOC that how to model the application software oriented the services, Service Oriented Modeling (SOM). Requirement modeling is the premise of the whole software development, so SOM is the first task of SOC[2].

In the semantic description of web service, Environment Ontology not only described the function of service, but also brought the domain knowledge in service description[13,17], which described the static semantics of service well. While the service is a kind of executable resource, in order to execute and composite intelligently, it is necessary to provide computer an apprehensible process of the inter-operation between the services. It is the dynamic description of service, for example, state machine[3~5], process algebra[6,7], and Petri net[8].Petri net is a well founded process modeling technique that have formal semantics. It presents a discrete parallel system as a direction graph, which is good at describing the sequence, alternative, conflict and synchronism.

Based on some earlier exploratory works[9], this paper references the relative concepts of Environment Ontology and Petri Net, and models the requirements as a Petri net. In this approach, environment ontology is expected to specify the capability of the Web services, and Petri Net is used to control the logic process.

The rest of this paper is structured as follows: Section 1 gives the notions of Intention and Task which is based on Environment Ontology (EO). Service composition modeling is presented as a Petri net. In section 3, a reachability tree is built from a simplified model. All the feasible service compositions are found through the algorithm similar as depth-first search. An example of travel arrangement is presented in Section 4 to verify the rationality of algorithms put forward above in this paper. Section 5 contains the concluding remarks.

II. THE REQUIREMENT MODEL OF SERVICE COMPOSITION

A. *Environment Ontology*

Web service is abstracted as an environment entity in EO, which intergrades with the specific domain knowledge. The environment entities[10] can be sorted to three classes: the causal entity, the autonomous entity and the symbolic entity. A hierarchical state machine is attached each causal entity for displaying its dynamic characteristics.

Definition 1 Environment Ontology structure is depicted as EO= {Ent, R, rel, HSM, EXCH, res}, in which:

Ent is a finite set of environment entities, and an entity =<id, Attr, Rans, values>,in which id is the name of an entity, Attr is a attribute vector, Rans is a vector of an attribute, values: Attr→Ran is a relation.

R is the set of rel.

rel: r→Ent×Ent, r is a relation function between the entities.

HSM is a finite set of tree-like hierarchical state machines.

EXCH ⊆ HSM×HSM is a message exchange relation between HSMs.

res: Ent ↔ HSM is a bijective relation. $\forall$ ent ∈ Ent, there is one and only one hsm ∈ HSM, such that hsm =res (ent).

Published by Atlantis Press, Paris, France.

More detail about Environment Ontology can be found in inference [11, 12].

B. *Intention*

Described by EO, the requirements of the user, could translate the initial state to the target state. If an entity could change the state from one to the other through an input from a service and give some outputs, we say that this service can fulfill the intention of this entity. Now we give the notion of Intention.

Definition 2 Intention: Let CE is a set of causal entities, $\forall$ r $\in$ CE, r:e =(s_o, s_t, I, O), in which:

- $s_o \in$ r.HSM is an initial state;
- $s_t \in$ r.HSM is the goal state;
- I $\in$ r.HSM is a finite set of inputs, which supply the entity some input messages;
- O $\in$ r.HSM is a finite set of outputs, which represents the effect of the intention.

And we signify r:(e_1,e_2,…,e_n) that there are n intentions on the entity r.

From the definitions above, it is easy to find that:

- These definitions are based on EO which gives the concepts of environment entity, state, input/output and hierarchical state machine.
- s_t must be the reachable state of s_0, or it makes none sense.

We consider the requirements as a number of intentions on several entities, between which there is some relation. Next this kind of relation is defined.

Mostly the intention has no directed effect on the others, called standalone intention. But some of them have a link. For example, as for Order:Paid(an intention "Paid" on the entity "Order"),there is a effect on Credit:Pay(an intention "Pay" on the entity "Credit"). It is not hard to know, both of them are met or not at the same time to maintain consistency.

Definition 3 Associated Intention: Let CE is a set of causal entities, $\forall$ r_1,$r_2 \in$ CE, $\exists$ r_1:e_i, r_2:e_j ,we say that r_1:e_i is associated with r_2:e_j ,if both of them muse be met or not at the same time. And we record the relation as A(r_1: e_i, r_2:e_j) or r_1: e_i A r_2:e_j.

It is not difficult to verify that the relation A reflexive and the symmetric, but not the transitive. For instance, in a travel arrangement case, Air_Ticket: PaidACredit_Pay, similarly, Train_Ticket:PaidA Credit_Pay. If the relation is transmissible, Air_Ticket: PaidATrain_Ticket:Paid. While choosing transportation, the tourist travels either by plane or train. It means that only one intention of them(Air_Ticket: Paid, Train_Ticket:Paid) needs to be satisfied. So the relation A is not transitive.

Definition 4 Let E be a set of Intentions. We define a binary relation R on E as <R>, which delegates R to be a compatible relation[18], if:

- R$\subseteq$E×E;
- R is reflexive;
- R is symmetric.

From this definition, we know that the relation A is <A>.

C. *Task*

According to the compatible relation on E, we divide the huge set of intentions to several sets, and then consider each son-set as a unit satisfied by a service.

Definition 5 Task: Let A $\subseteq$E×E and <A>, E is the set of intentions, then t_i=E_i, if:

(1) $E_i \subseteq E$;

(2) $\forall$ $e_m \in E_i$, $\forall$ $e_n \in E_i \rightarrow A(e_m,e_n)$;

(3) $\neg \exists$ $e_p \in (E-E_i)$, $\forall$ $e_m \in E_i \rightarrow A(e_p,e_m)$.

Example let E ={e_1,e_2,e_3,e_4,e_5},the relation A={(e_1,e_2), (e_1,e_4),(e_2,e_4),(e_2,e_3),(e_2,e_5), (e_2,e_1), (e_3,e_5), (e_3,e_2), (e_4,e_1), (e_4,e_2), (e_4,e_5), (e_5,e_2),(e_5,e_3),(e_5,e_4), (e_1,e_1), (e_2,e_2),(e_3,e_3), (e_5,e_5),(e_4,e_4)}.

It is easy to verify that the relation A is <A>, then we could write it in the way that <A>={(e_1,e_2), (e_1,e_4),(e_2,e_4),(e_2,e_3),(e_2,e_5),(e_3,e_5),(e_4,e_5)}. Furthermore, E_1={e_1,e_2,e_4},E_2={e_2,e_3,e_5},E_3={e_2,e_4,e_5}, and all of them are the tasks of E. Let E_1 be an example, first, $E_1 \subseteq E$; second, (e_1,e_2) $\in$ A, (e_1,e_4) $\in$ A,(e_2,e_4) $\in$ A; and then E-E_1={e_3, e_5}, (e_1,e_3) $\in$ A,(e_1,e_5) $\in$ A. So t_1=E_1.

To understand conveniently, let us observe the figure of <A>$\subseteq$E×E as following.

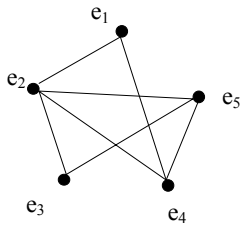

Figure 1 The figure of <A>

In a general way, under a contract condition, it is concise and unambiguous to use a figure instead. In the light of the concept in set theory[18], we have some method to get the Tasks from E :

(1) The set of nodes on any complete polygons is a task;

(2) An independent node is a task;

(3) The set of nodes on any side of incomplete polygons is a task.

To this point, we have classified the intentions as tasks, where there is only an intention or maybe several intentions. As for service composition, a task is viewed as an atomic unit, which is contented by a single service.

D. *The requirement model of service composition*

The service composition requirement is respected as a Petri Net, which is a visual and graphical modeling technique.

Definition 6 A service composition requirement is a marked Place/Transition net, SCR= (P, T; F, M_0, M_t) where

- P is a finite set of places;
- T is a finite set of tasks(transitions in Petri net) ;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of direction arcs(flow relation);
- M_0 is the initial mark;
- M_t is the final mark.

The capacity of places in SCR is one, and the weight of the arcs is one, so SCR is an elementary net system.

III. SERVICE COMPOSITIONS OF SCR

Now we consider the SCR reachability problem, whether an arbitrary state can be reachable in the model, or whether there is some reasonable effective description of the set of all reachable states[13]. The former is the base of many others, since many operations research problems can be defined through it.

In fact, it is not practical to explore the net exhaustively because of the famous problem of combinatorial explosion: the size of the state-space may grow exponentially with the size of a model. So we need simply the net first to limit this explosion.

A. Simplify the SCR

The model of SCR consists of the four parts: place, task, directed are, token. Places represent passive components which store "tokens" and take particular states. Tasks represent the active components which may produce, transport or change "tokens". When the time of transporting tokens is not taking into account, the model can be simplified. On the basis of this method, we have the rule of absorbing places as follow[14,15]:

The places, which have only one input and output inter-place, can be absorbed, and the tasks are combined to tasks in its higher level.

Here is the sketch of this rule. As depicted in figuration 2a and 2b, the place P_j meets the rule above, so it is absorbed and task t_n is combined to t_m.

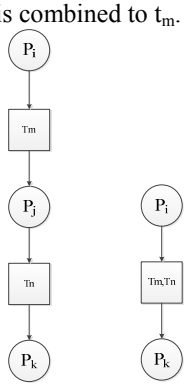

Figure 2a The primary model Figure 2b The simplified model

After absorbing places, there are some tasks merged together, and then we define this kind of tasks as linear task<t_m,t_n> in figure 2b.

Making use of the rule, the places and tasks are reducing. The scale of the model becomes concise, so it gets easily to find the tasks firing sequences, namely service compositions; meanwhile this measure cuts down on the calculated amount.

B. Find all the possible service compositions

The behavior of SCR can be described in terms of system states and their changes. In order to simulate the dynamic behavior of it, a state or marking in a Petri nets is changed according to the following transition (firing) rule[15]:

(1) $\forall$ t ∈ T, if $\forall$ p ∈ •t, M(p) ⩾ 1, then we say that M enable t or t is enable. And we denote it as M [t>;

(2) If M enable t to change itself to be a new one as M′ , then

$$M'(p) = \begin{cases} M(p)-1, & p \in \bullet t - t\bullet \\ M(p)+1, & p \in t\bullet - \bullet t \\ M(p), & \text{others} \end{cases}$$

And we denote it as M [t>M′ .

Definite 7 Let CSR=(P,T;F,M_0,M_t) is a Petri net model, if $\exists$ t∈T, M[t>M′ , then we say M′ is reachable from M.

If $\exists$ $t_1, t_2, \ldots, t_k$, and $\exists$ $M_1, M_2, M_3, \ldots, M_k$,

$M[t_1>M_1[t_2> M_2 \ldots M_{k-1}[t_k>M_k(1)$,

then we say M_k is reachable form M. We denote the tasks sequence $t_1t_2t_3\ldots t_k$ by σ , accordingly the expression (1) can be written as M [σ > M_k. The sequence like σ , is a service composition leading to the marking M_k.

To find all the service compositions consist of two parts. First, a reachable tree can be built according to the simplified model[15]. Second, by depth-first search, all possible solutions (service compositions) can be found in the form of reachability tree.

Algorithm 1 Algorithm for constructing the reachability tree of SCR

Input: CSR= (P, T, F, M0, Mt) ;

Output: RT(SCR);

Step1: Make use of the rule of absorbing places; simplify the initial model CSR;

Step2: Take M_0 as the root node of RT (CSR) and mark it with "new";

Step3: While there is a "new" node, Do

let any "new" node be M;

Step4: If there is a node M in the directed routing from M_0 to M, Then

change the marking "new" to "old", Run to Step3;

Step5: If M=Mt Then

let M be a leaf, Run to Step3;

step6: For $\forall$ t ∈ T , if M[t> Do

(1) get the new marking M′ ,after M[t>M′ ;

(2) add the new node M′ to RT (CSR), draw a directed arc from M to M′ ; write t beside the arc; omit "new", run to Step3,

The transitions of markings and tasks occur alternately. The algorithm is terminable proved in [16].

In this reachability tree, the amount of leaves is the amount of service compositions. Next through searching the reachable tree, we could get all the possible service compositions.

Algorithm 2 Algorithm for all the service compositions

Input：RT(SCR)；

Output：T

Step1：Visit the root of RT(CSR), and mark the node with "visited";

Step2：While there is a "visited" node, Do

Depth-first search its son-node. Mark it with "visited", and output the tasks besides the arc;

Algorithm 2 is a recursive algorithm used to output the reachable paths, the so-called service composition.

IV. CLASSIC CASE

A. *Illustration of the case*

It is a travel arrangement case. A tourist wants to take a trip to scenic spots A, and then goes to spots B. He could get the city of scenic spot A by plane or train. If he takes a plane, the ticket needs to be delivered to him. Meanwhile a hotel is booked ahead around spot A. From the scenic spot A to B, he could take buses or trains. Similarly, a hotel is booked ahead with advanced deposits. All the cost can be paid by credit-card.

On the basis of Environment Ontology, the intentions on the entities can be indicated: Air_Ticket:(Book, Paid, Deliveried), Train_Ticket:(Book, Paid), Bus_Ticket: (Book, Paid), Hotel_Room: (Book, Paid), Credit_Card: Pay, Express:Delivary.

Taking the association relation into account, we can define the tasks in this case as follows;

Table 1 The tasks in travel arrange

Task	***Expression***
t_1	{Air_Ticket: Book}
t_2	{Air_Ticket: Paid, Credit_card: Pay}
t_3	{Air_Ticket: Deliverid, Express: Delivary}
t_4, t_7	{Train_Ticket: Book}
t_5, t_8	{Train_Ticket: Paid, Credit_card: Pay}
t_9	{Bus_Ticket: Book}
t_{10}	{Bus_Ticket: Paid, Credit_card: Pay}
t_6, t_{11}	{Hotel_Room: Book}
t_{12}	{Hotel_Room: Paid, Credit_card: Pay}

Integrated the tasks on the table over, the travel arrange model can be given as Fig. 3:

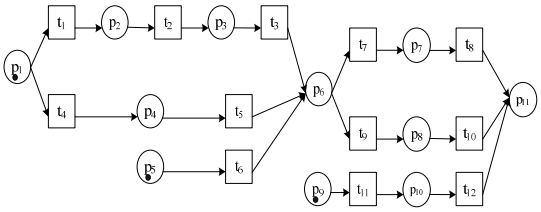

Figure 3 The Petri net model of CSR_TravelArrange.

B. *Find all the service compositions*

First, we simplify the model of CSR_TravelArrangement; and the new model is shown as Fig. 4:

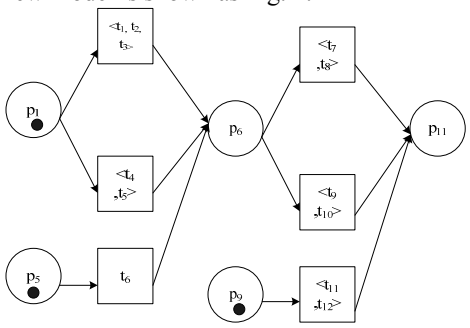

Figure4 The simplified model of CSR_TravelArrange

Then we build the reachability tree as steps as following:

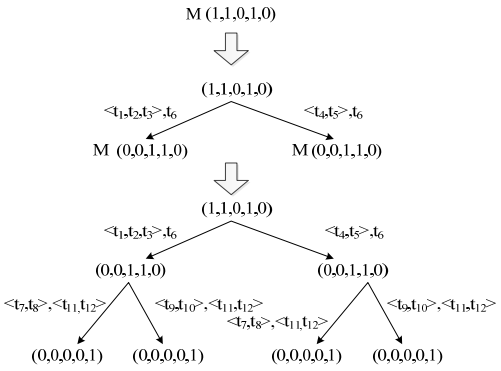

Figure 5 The course of building the reachability tree

From this reachability tree, we know that there are four service compositions because the reachability tree has four leaves. Then algorithm 2 gives all the for compositions: {t1, t2, t3, t6, t7, t8, t11, t12}, {t1, t2, t3, t6, t9, t10, t11, t12}, {t4, t5, t6, t7, t8, t11, t12}, {t4, t5, t6, t9, t10, t11, t12}.

V. CONCLUSIONS

In this paper, we proposed a method to find all the service compositions of the requirement model. First, based on Environment Ontology, we modeled the requirement of service compositions as a dynamic net system, named a Petri net. After simplified, the SCR model is constructed to be a reachability tree. At last, through depth-first search, we get all the compositions. The travel arrangement example verified the effectiveness of our method.

There are some other issues we will research forward:

Published by Atlantis Press, Paris, France.

(1) When we analyze the SCR model, there are some other operations, such as insertion, substitution, decomposition and composition.

(2) Introducing QOS into the SCR, it can be used to service composition selection. For example, we could use the insertion operation, and introduce QOS into the SCR model as a component of a Petri net.

ACKNOWLEDGMENT

The Natural Science Foundation of Zhejiang Province of China under Grant No. Y1110483.

REFERENCES

[1] Papazoglou Mike P. Service-oriented computing: concepts, characteristics and directions[C]// Proceedings of the fourth international conference on web information systems engineering. Roma, Italy, 2003:3-12.

[2] Wu Budan, Jin Zhi, Zhao bin. Service-oriented modeling based on whole process asset reuse[J]. Chinese journal of computers,2008, 31(8): 1293-1308.

[3] Andreas W, Peter F. Matchmaking for business process based on choreographies [J]. IEEE on E-Technology, E-Commerce and E-Service, 2004:359-368.

[4] Andreas W. Transforming BFEL into annotated deterministic finite State automata for service discovery [J]. IEEE on Web service, 2004:316-323.

[5] Andreas W. IPSI-PF: A business process matchmaking engine [J]. Information systems and E-business management, 2004,3(2):127-150.

[6] Gwen S. Describing and reasoning on Web service using Process Algebra [J]. IEEE on Web services, 2004:43-50.

[7] Andrea F. Web Service: A Process algebra approach[J].ICSOC'04 Proceedings of the 2th international conference on Service oriented computing,2004:242-251.

[8] Peter C. A., Kommunikation mit Automaten. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. In German. Also in: New York: Griffiss Air Force Base, Technical Report RADC-TR-65-377, vol.1, pages 1-Suppl.1. 1966.

[9] Ye Ronghua, Jin Zhi, Zhong Farong. Requirement mode and satisfiability decision for service compositon[J]. Journal of frontiers of computer science and technology,2011,5(5):458-466.

[10] Wang Puwei, Jin Zhi, Liu Hongyan. The description and found of E-commerce software entity [J]. Science in China press,2009,39(12):1271-1287.

[11] Wang Puwei, Jin Zhi, Liu Lin, et al. Building toward capability specifications of Web services based on an environment ontology[J]. IEEE Transactions on Knowledge and Data Engineering, 2008,20(4):547-561.

[12] Wang Puwei, Jin Zhi, Liu Hongyan. The function description and its findings of Internet ware entities[J].Science in China: Series F Information Sciences,2009,39(12):1271-1287.

[13] Ernst W. Mayr. An algorithm for the general Petri net reachability problem[J].Society for industrial and applied mathematics,1984,13(3):441-461.

[14] Liu T S, Chiou S, B. The Application of Petri Net to Failure Analysis[J]. Reliability engineering and system safety, 1997,57:129-142.

[15] Wu Zhehui. Introduction of Petri Net[M].Beijing : China machine press，2006.

[16] Peterson J.L. Petri net theory and system theory[M].Xuzhou : China university of mining and technology press，1989.

[17] Cai Guangjun, Jin zhi. Based on Environment Ontology web service description: a projection approach[J]. Computer science，2009,36（8）：116-120.

[18] Wu Tongjia, Xiao xi'an. Introduction of set theory[M]. Dalian : Dalian university of technology press，2008.