BestRec: A Behavior Similarity based Approach to Services Recommendation

Zicheng Huang, Jinpeng Huai, Hailong Sun, Xudong Liu, and Xiang Li
Beihang University, Beijing, China
{huangzc, huaijp, sunhl, liuxd, lixiang}@act.buaa.edu.cn

Abstract

Development efficiency is an important factor for the Internet-scale software produced through services composition. In this paper, we aim at improving development efficiency from two aspects. One is to improve services discovery efficiency through recommendation, and the other is to provide a mechanism to reuse an existing composite service. We propose BestRec, a behavior similarity based approach to services recommendation. With BestRec, the behavior captured by a composite service is described by a NFA-based service model. Based on the service model derived from the developing context of services composition and the existing composite services in the services repository, we calculate the behavioral similarity according to the relations between states of each existing service model and that of the developing service model. Through this approach we can automatically model the request of services discovery and recommend the most suitable composite services to developers in the light of their behavioral similarity. Finally, we illustrate the effectiveness of BestRec with a case study and experimental evaluation in the area of Web services testing applications.

1. Introduction

In open network environments, software requirements, which are diverse and dynamically changing, are difficult to meet with traditional software development methods. With service oriented technologies like SOA architecture and Web services, users can quickly develop software by composing available services [1].

Currently, process-based services composition methods are drawing more and more attention, which are based on the traditional workflow and business process management, and they have been used in many industrial business applications [2]. Similar to the traditional software development, how to create new services with high-efficiency, rapid-development, and low-cost is an important goal for service composition. The development of composite service still faces some challenging issues that limit the improvement of efficiency. Firstly, with the wide application of Web services and SOA technologies, the number of existing accessible and reusable services increased rapidly, and the service description is becoming more complicated. Developers need to learn sufficient service description standards, query technologies and domain knowledge to discover appropriate services for composition. This limits the degree of automation for services composition, thus an active services recommendation mechanism will help to achieve that goal. Secondly, existing services discovery algorithms mainly focus on the syntax, semantic or structural relations of services. But we claim that the function of a composite service mainly conforms to its behavior. Two services with high syntax, semantic and structural similarity may perform rather different behaviors. The lack of behavioral relations has greatly influenced the accuracy of services discovery. At the same time, some works in [3,4] have also emphasized the importance of behavior information as a factor for services retrieval and discovery.

On the other hand, we have noted that besides single service reusing, some business process segments can also be reused. In some specific domains, after a long-term practical operation, a number of operational activities and business processes have been accumulated and formed standard processes. These standard processes contain useful information that is helpful for the development of new services. Therefore it is effective and feasible to consider process as an abstract granularity for services reusing. Meanwhile, process-based composite services allow the requirement of services discovery to be described by an abstract process model, which can be automatically captured during the processing of services composition.

In this paper, we propose BestRec, a behavior similarity based approach to active composite services recommendation. With BestRec, the *Non-Deterministic Finite State Automata* (NFA) is used to describe the dynamic behaviors of composite service. And we design algorithms for behavior similarity measurement based on simulation of the states in the NFA-based model. Based on these algorithms, the approach of recommendation is discussed and evaluated.

The rest of this paper is organized as follows. Section 2 addresses a motivation scenario for composite service development, and introduces the basic idea of BestRec. We describe a NFA-based model for composite service description, and define the requirement of services discovery as a developing composite service in section 3. Section 4 introduces the algorithms for composite service similarity measurement according to behavioral features and discusses the approach of services recommendation. The architecture and implementation of our service recommendation system is presented in section 5. In section

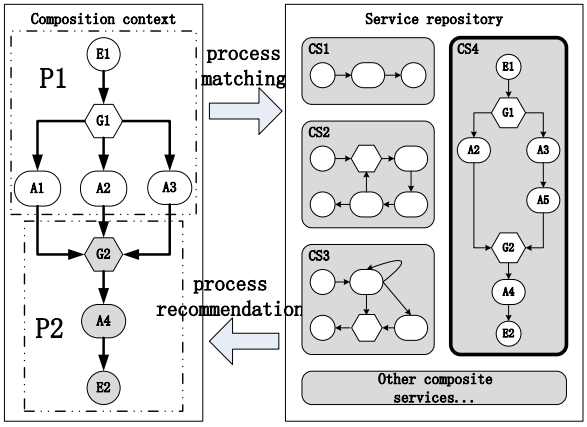


Figure 1. Scenarios of services recommendation in the process-based services composition

6 the effectiveness of our approach is evaluated by a case study. There are some related researches on processes matchmaking and services recommendation in section 7. Finally, Section 8 concludes our work.

2. Motivating scenarios

For process-based services composition, developers usually use process modeling languages such as BPMN [5] for services orchestration. In each stage of orchestration, the finished composite services have a strong flow characteristic, and can be used as reference of services discovery to meet the requirements. As shown in Figure 1, the left part is the context of services composition, including information of process structure, event, activities, *etc.* (dotted P1 part), which can be extracted by process modeling tools. The right part of Figure1 shows the available composite services maintained in service repository.

In this paper, we discuss the services discovery and recommendation for the reusable composite services. CS1, CS2, CS3 and CS4 define the process structure, event and activities of composite services in service repository. We match them with the completed fragment in the context (P1), and then find out the CS4 with highest similarity, thus CS4 is recommended to developers through process modeling tools. Developers can rapidly finish development according to the information in CS4 with a few modifications (dotted P2 part). Comparing with the directly development of fragment P2, the development efficiency for whole composite service is improved.

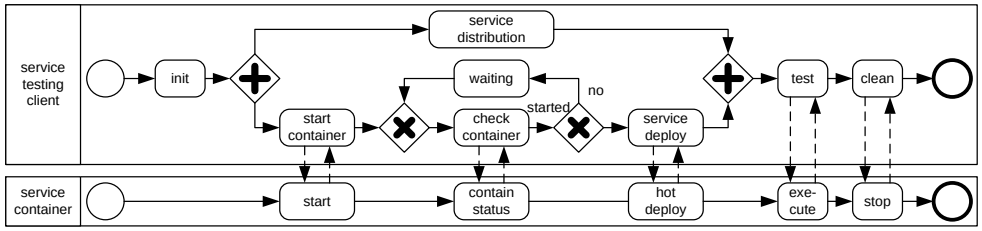
In order to achieve the composite services recommendation mentioned above, first of all we need to define a model, which is suitable to describe the process structure, events, and activities for the current composite services and is also appropriate for similarity determination of behavior. On the other hand, similarity algorithms should support quantitative similarity measurement between composite service and its fragments.

3. Composite service description

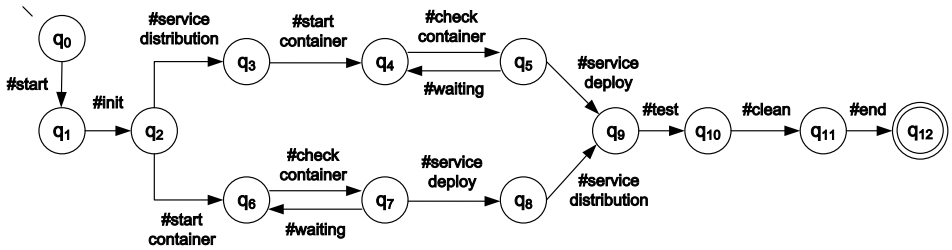
The dynamic behavioral characteristics of composite service always include its internal control flows, data flows, interactive protocols and state transitions [6]. In this paper we define a service model to describe the static and dynamic information of composite service.

3.1. Composite service description model

The behavior of composite service is comprised of the states of its inner process, interaction protocols with other services and the sequence of invoked activities during its execution, *etc*. So as to realize the similarity measurement



(a) Inner process of a Web services testing application



(b) *NSM* model of the test client

Figure 2. Inner process of composite service and the corresponding *NSM* model

of composite services according to their behaviors, the description of dynamic behavioral characteristics of composite service are needed. We use a service description model based on NFA to satisfy this requirement. The formal definition of this description model is showed in Definition 1.

Definition 1 (NFA-based service model). *A NFA-based service model NSM is represented as a tuple NSM = ($\boldsymbol{Q, \Sigma, \delta, q_0, F}$) where :*

- $\boldsymbol{Q}$ *is a finite set of states,*
- $\boldsymbol{\Sigma}$ *is a finite set of activities in the modeled service,*
- $\boldsymbol{\delta : Q \times \Sigma \rightarrow Q}$ *represents the state transitions of the service inner process,*
- $\boldsymbol{q_0}$ *is a start state with* $\boldsymbol{q_0 \in Q}$*, and*
- $\boldsymbol{F \in Q}$ *is the set of final states.*

Figure 2 shows an inner process of composite service and its corresponding *NSM* model. The process displayed in Figure 2(a) is the inner process of a Web services testing application which is modeled using BPMN. There are two participants included in this process, the upper part of describe the process of test client, including initialization, service deploy and service testing *etc*. The bottom part describes the behavior of Web services container. Figure 2(b) is the corresponding *NSM* model of the test client.

In order to achieve the purpose of real-time services recommendation (as shown in the motivating scenarios) during the development of composite service, the behavioral characteristics of the developing composite service are needed to be described. We use the *NSM* model to describe the developing composite service as a process segment. Since the composite service is under developed, the crucial problem of the description is to determine the final states of the process segment. We firstly cut of sequence flows without output object if exist. Then the events or activities without any output sequence flows will be considered as the pre-final activities and a sequence flow from each of them to an additional end event will be added.

3.2. Mapping from BPMN model to *NSM*

Currently, BPMN has become one of the mainly used modeling languages during services composition. In this paper, we use BPMN as an example to explain the conversion between business process model and the *NSM* model. To simplify the problem, we only discuss a core subset of BPMN. Definition 2 gives a formal definition of the core BPMN process.

Definition 2 (Core BPMN process). *A core BPMN process is a tuple CBP =* $\boldsymbol{(O, A, E, G, T, S, E^S, E^I, E^E, G^F, G^J, G^V, G^M, F)}$ *where:*

- $\boldsymbol{O}$ *is a set of objects which can be partitioned into disjoint sets of activities* $\boldsymbol{A}$*, events* $\boldsymbol{E}$*, and gateways* $\boldsymbol{G}$*,*
- $\boldsymbol{A}$ *can be partitioned into disjoint sets of tasks* $\boldsymbol{T}$ *and sub-process invocation activities* $\boldsymbol{S}$*,*
- $\boldsymbol{E}$ *can be partitioned into disjoint sets of start events* $\boldsymbol{E^S}$ *, intermediate events* $\boldsymbol{E^I}$ *, and end events* $\boldsymbol{E^E}$*,*
- $\boldsymbol{G}$ *can be partitioned into disjoint sets of parallel fork gateways* $\boldsymbol{G^F}$ *, parallel join gateways* $\boldsymbol{G^J}$ *, XOR decision gateways* $\boldsymbol{G^V}$ *, and XOR merge gateways* $\boldsymbol{G^M}$*,*
- $\boldsymbol{F \subseteq O \times O}$ *is the control flow relation, i.e. a set of sequence flows connecting objects.*

BPMN has few semantic constrains for modeling, so the composite services developed by BPMN have a wide diversity. At the same time, these arbitrary processes structures make it more complex to convert to the *NSM* model. Therefore, we add some constraints based on Definition 2 according to the method from [7]: 1) Two parallel flows initiated by a parallel fork gateway, should be joined by a parallel join gateway. 2) Two alternative flows created via a decision gateway, should be synchronized by a merge gateway. We denote the BPMN model which meets these features by Well-structured core BPMN process.

The mapping from the Well-structured core BPMN process to the *NSM* model includes two parts: basic elements mapping (Event, Task, Sub-process, Sequence flow) and control elements mapping (Sequence, Switch). During the basic elements mapping, a task or an intermediate event is mapped onto a state transition with one input state and one output state. The state transition, being labeled with the name of that task or event, models the execution of the task or event. A start or end event is mapped onto a similar module except that a start state with a state transition labeled with "Start" or a final state with a state transition labeled with "End" is used to signal when the process starts or ends. A sub-process may be treated as an independent BPMN process and be mapped straightforward. Generally, any sequence flow is mapped onto a state.

Process modeling languages usually support two basic process control structures: sequence and switch. Figure 3 presents the mappings between the two process control structures and the *NSM* model. Sequence is described by a set of sequential state transitions (as shown in Figure 3(a), and switch is described by uncertain state transition relations (as shown in Figure 3(b)).

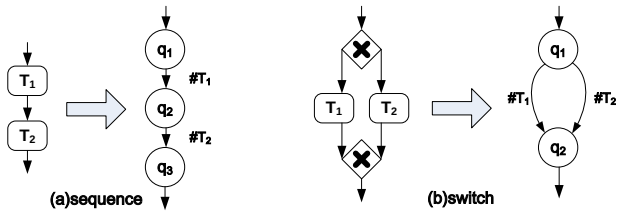


Figure 3. Mappings from control elements to *NSM*

4. Similarity measurement and recommendation of composite services

Based on the definition of *NSM* model, we propose a novel services discovery approach which can capture the behavioral characteristics of composite service and recommend to developers the most appropriate composite service to further conduct their development. Our approach includes two steps where firstly measuring the behavioral similarity of composite service. And finally determine the approach of recommendation according to the similarity information achieved.

4.1. Similarity measurement of composite services

The behavioral characteristics of composite service can be captured and described by a sequence of the activities being invoked. In the *NSM* model, we can use a sequence of state transitions from the start state to one of the final states to capture this information. So as to involve the whole behavioral characteristics of composite service in services recommendation, we should consider the influence of single activity's invocation to the whole sequence and combine them together. There are some researches concern about the structure of process (type and degree of the nodes in a process diagram.*etc*.) to measure the similarity of processes [8,9]. But we have noted that the function of a composite service mainly conforms to its behavior, and the structural and behavioral similarity is not necessarily related.

Our behavioral matching technique is based on the notion of bisimilarity between state-machines [10]. Two states are bisimilar if they can transit to bisimilar states via the same or similar transitions [11]. In order to find the most matching state set of two processes according to their behavioral similarity, we introduce an algorithm for computing a quantitative measuring value based on the approach of simulation-based graph similarity in [12, 13]. The algorithm recursively computes a similarity degree for each state pair (*s, t*) by measuring the similarity degrees of forward transition states of *s* and those of *t*, combining with the similarity of corresponding transitions. The process attaining the similarity of two states is enumerated between all the forwarding transitions of them.

We denote by *Sim*(*s, t*) the degree of similarity between states *s* and *t* and formalize the computation of *Sim*(*s, t*) as follows. Let $s \overset{a}{\Rightarrow} s'$ be a state transition from *s* to *s'* through an activity *a*, that is $\delta(s, a) = s'$. In addition, we define two similarity functions: a *node similarity function* $N : Q \times Q \rightarrow [0, 1]$, and a *label similarity function* $L : \Sigma \times \Sigma \rightarrow [0, 1]$. To find the best match for s among the forward neighbors of *t*, we need to maximize the value $L(a, b) \times Sim(s', t')$. The similarity between a final state and another state is equal to the node similarity of them.

Definition 4 (State similarity of two processes). *Let* $NSM_1 = (\boldsymbol{Q_1}, \boldsymbol{\Sigma_1}, \boldsymbol{\delta_1}, \boldsymbol{q_{01}}, \boldsymbol{F_1})$ *and* $NSM_2 = (\boldsymbol{Q_2}, \boldsymbol{\Sigma_2}, \boldsymbol{\delta_2}, \boldsymbol{q_{02}}, \boldsymbol{F_2})$ *be two processes of composite services. The states similarity function of them is:*

$$Sim(s,t) = \begin{cases} N(s,t) & if \forall a, \nexists \delta_1(s,a) \in \boldsymbol{Q_1} or\ s \epsilon \boldsymbol{F_1} \\ W_1 + \frac{p}{n} \cdot W_2 & otherwise \end{cases}$$

where

$$W_1 = (1-p) \cdot N(s,t),$$
$$W_2 = \sum_{s \overset{a}{\Rightarrow} s'} \max_{t \overset{b}{\Rightarrow} t'} \left(L(s,t) \cdot Sim(s',t') \right),$$

and p is the important factor between local similarity and the similarity of successor states, n is the number of transitions leaving s.

4.2. Composite services recommendation

In our motivating scenarios, the function *Sim*(*s, t*) indicates the corresponding states of the composite services. We then use an approach based on linear averages to evaluate the similarity of two processes as a whole.

Definition 5 (Processes similarity of two composite services). *Let* $NSM_1 = (\boldsymbol{Q_1}, \boldsymbol{\Sigma_1}, \boldsymbol{\delta_1}, \boldsymbol{q_{01}}, \boldsymbol{F_1})$ *and* $NSM_2 = (\boldsymbol{Q_2}, \boldsymbol{\Sigma_2}, \boldsymbol{\delta_2}, \boldsymbol{q_{02}}, \boldsymbol{F_2})$ *be two processes of composite services. The process similarity function of them is ProSim:* $(NSM \times NSM) \rightarrow [0, 1]$ *that:*

$$ProSim(NFM_1, NFM_2) = \frac{\sum_{s \epsilon \boldsymbol{Q_1}} \max_{t \epsilon Q_2} Sim(s,t)}{|\boldsymbol{Q_1}|}.$$

We use a threshold σ to determine whether a composite service should be recommended to developers. The users can review and, if necessary, adjust this threshold according to their experiences. The composite services, whose degree of similarity with the developing service is higher than σ, are finally recommended during services composition. Let n_1 and n_2 be the number of states in the input *NSMs* of *ProSim*. While the calculation begins with final states, the time complexity of *ProSim* is $O(n_1 \times n_2)$ as it needs to check $n_1 \times n_2$ state pairs.

5. System design and implementation

This section introduces the design and implementation of our system, including its architecture, functional modules and implementation details. We implemented our services recommendation approach BestRec in the SOArWare environment, which is a service oriented software production line. SOArWare includes a BPMN modeler for BPMN business models orchestration and a service repository for services storage and management.

In SOArWare the existing web services are collected by means of the service repository and two service discovery modes are supported. One mode is querying the service repository by keywords or query language.

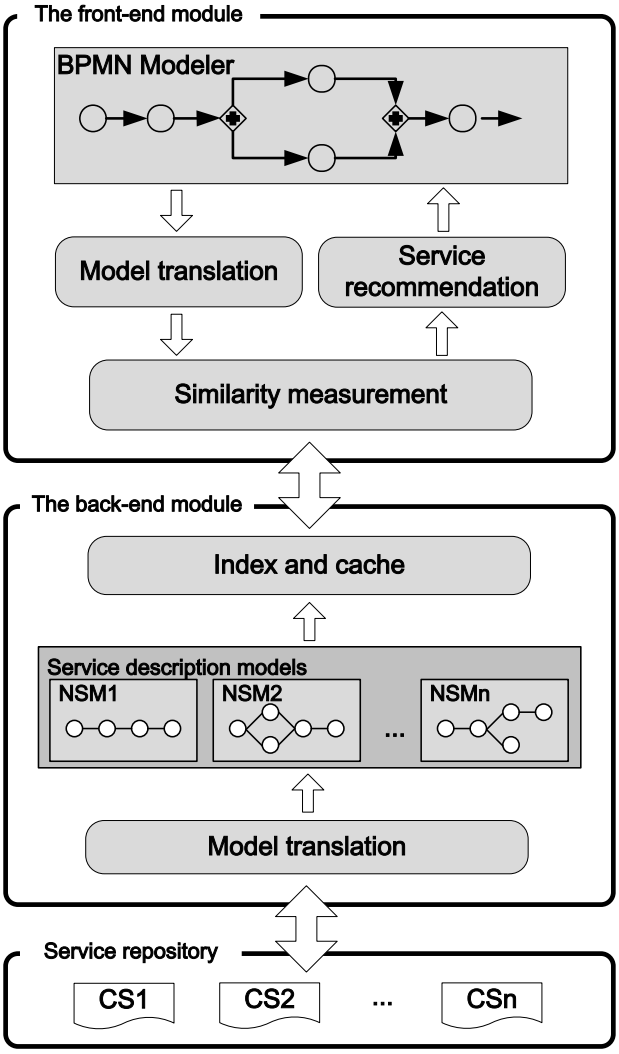


Figure 4. System architecture of BestRec

The other is recommending services actively to developers during the course of the development. The last mode is realized by BestRec. Figure 4 shows the system architecture of BestRec.

As shown in Figure 4, the BestRec system is divided into two modules. One is the back-end module and the other the front-end module. The back-end module of BestRec is running according to the change of service repository and the front-end module. Especially when a new service has been developed and registered to the repository or a new service has been collected from Internet. The model translation tool maps a composite service business process model into our composite service description model(as shown in section 3). It accepts a BPMN model as input data and creates its corresponding *NSM* model as output data. The index and cache tool makes an index for each *NSM* and caches the latest similarity degree between the developing service and each *NSM*. This tool can increase the query efficiency of *NSMs*

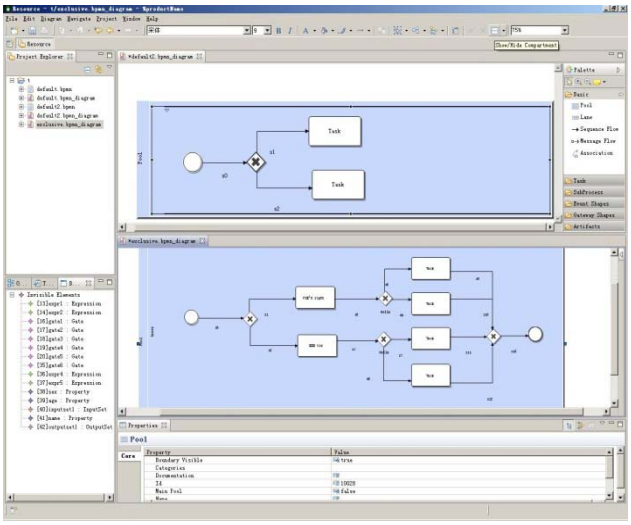


Figure 5. BPMN Modeler with services recommendation

and support the roll-back selection of recommended services. The front-end module of BestRec is integrated with the BPMN modeler of SOArWare. During every change of the developing composite service business model, the model translation tool automatically maps the unfinished business model to a *NSM*. Then the similarity measurement tool is activated and continuously calculates the similarity degree between the *NSM* of developing service and that of each service available from service repository using *ProSim*. The service recommendation tool ranks available services according to their similarity degree between the developing service. Developers can assign a threshold for service recommendation, available service whose similarity degree is greater than the threshold will be recommended to the BPMN modeler.

Figure 5 is a screenshot of the SOArWare BPMN modeler integrated with the BestRec system. The upper mid part of the modeler is the main editor area of BPMN orchestration and the lower left part is the recommended services list. This list performs real-time refresh according to each change of the developing service business model. When a recommended service in the list is selected, the lower mid area of the modeler will display the BPMN model of the selected service and the property and parameter details of the model will be presented below it.

6. Case study and experiment

In this section, we illustrate the effectiveness of BestRec by a case study of developing a services testing application. Furthermore, we evaluate *ProSim* by measuring the accuracy of the state simulation relations it produces, and the appropriate value of threshold is discussed according to the evaluation.

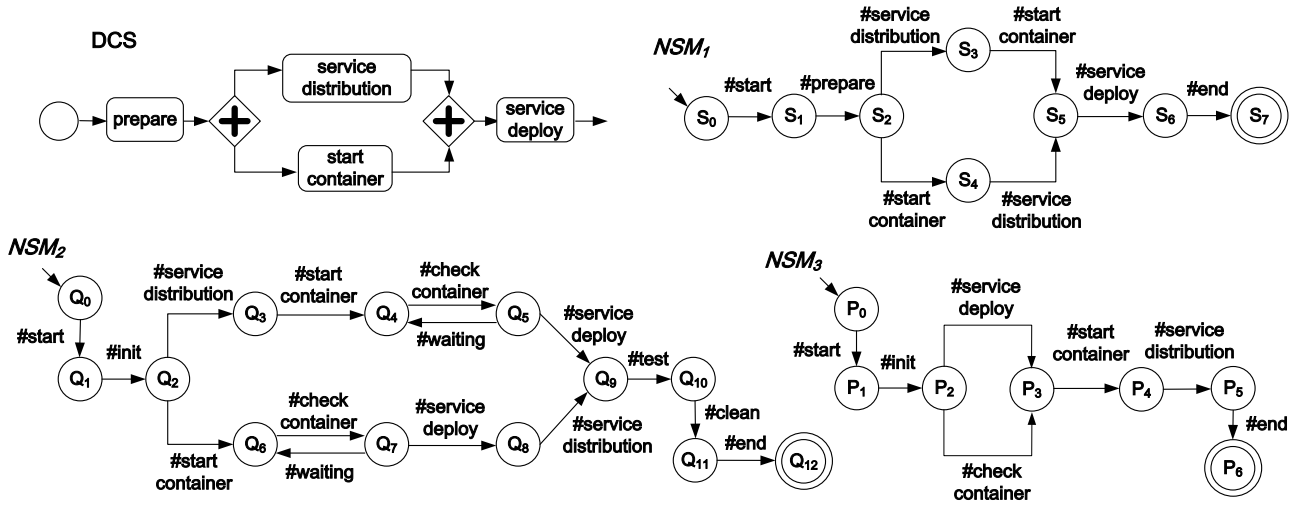


Figure 6. Evaluating scenario

We conduct an evaluating scenario which includes three *NSMs* to be matched. As shown in Figure 6, DCS is a developing composite service in the BPMN modeling tool. NSM_1 is the service model mapped from DCS and the other two are models of composite services in the service repository. We compute the degree of behavior similarity between NSM_1 and the other two models and then recommend to developers the one with higher degree. In addition, we compare our approach with the structural similarity based service matchmaking algorithm presented in [14] to show the effectiveness of our work.

We derive the correct relations between the three *NSMs* according to the domain experience of Web services development by considering the forward and backward transitions of two states. The result and some characteristics of these models are showed in Table 1, where *S* is the number of states and *T* is the number of transitions of the model. Through this result, we can see that in our scenario NSM_2 is more similar with NSM_1 than NSM_3 and should be recommended.

Table 1. Characteristics of the evaluated *NSMs*

NSM_1		NSM_2		NSM_3		Correct relations	
S	***T***	***S***	***T***	***S***	***T***	NSM_1 & NSM_2	NSM_1 & NSM_3
8	8	13	15	7	7	7	5

The states similarity is calculated through *ProSim* and the results are showed in Table 2 and Table 3. Table 2 shows the states simulation relations between NSM_1 and NSM_2, while Table 3 shows the relations between NSM_1 and NSM_3. The bold and italic values in the tables indicate the corresponding states between *NSMs*.

Table 3. States simulation between NSM_1 and NSM_3

	P_0	P_1	P_2	P_3	P_4	P_5	P_6
S_0	***0.87***	0.47	0.39	0.42	0.5	0.5	0.0
S_1	0.52	***0.75***	0.55	0.57	0.67	0.67	0.67
S_2	0.41	0.57	***0.67***	0.61	0.65	0.57	0.5
S_3	0.5	0.66	0.58	***0.73***	0.67	0.67	0.67
S_4	0.5	0.66	0.57	0.6	***0.78***	0.67	0.67
S_5	0.41	0.56	0.63	***0.69***	0.65	0.57	0.5
S_6	0.5	0.64	0.55	0.59	0.75	***1.0***	0.67
S_7	0.0	0.67	0.5	0.5	0.67	0.67	***1.0***

Table 4. Structural similarity of *NSMs*

	CPC	LCStr	LCSeq	ED	Comb
NSM_2	0.59	0.21	0.71	0.36	0.47
NSM_3	0.75	0.25	0.59	0.5	0.52

Table 2. States simulation between NSM_1 and NSM_2

	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}
S_0	***0.91***	0.47	0.41	0.47	0.39	0.39	0.39	0.41	0.48	0.42	0.5	0.5	0.0
S_1	0.54	***0.82***	0.56	0.64	0.56	0.54	0.57	0.56	0.64	0.57	0.67	0.67	0.67
S_2	0.42	0.61	***0.85***	0.61	0.49	0.64	0.51	0.69	0.61	0.5	0.57	0.57	0.5
S_3	0.5	0.68	0.73	***0.83***	0.56	0.57	0.56	0.57	0.67	0.57	0.67	0.67	0.67
S_4	0.50	0.67	0.68	0.66	0.56	0.57	0.56	0.61	***0.84***	0.57	0.67	0.67	0.67
S_5	0.41	0.54	0.5	0.56	0.66	0.63	0.66	0.66	0.57	***0.69***	0.65	0.57	0.5
S_6	0.5	0.64	0.57	0.63	0.54	0.55	0.54	0.57	0.65	0.59	0.75	***1.0***	0.67
S_7	0.0	0.67	0.5	0.67	0.5	0.5	0.5	0.5	0.67	0.5	0.67	0.67	***1.0***

Through these values, we get the similarity measurement where $ProSim(NSM_1,NSM_2)$ = 0.87 and $ProSim(NSM_1,NSM_3)$ = 0.81. Table 4 shows the structural heuristics of NSM_1 and the other two models according to [14]. These heuristics are *common process count* (CPC), *longest common substring* (LCStr), *longest common subsequence* (LCSeq) and *edit distance* (ED). Comb is the linear combination of these heuristics. As a result, the composite service of NSM_2 will be recommended in our approach, while in structural measurement that of NSM_3 may be recommended.

7. Related work

Research work related to our paper can be classified into two categories: (1) similarity of business process models and (2) services discovery and recommendation.

Similarity of business process models. There are a few researches that address the problem of evaluating similarity of business processes. Process variants [15] is an approach to derive similarity by deciding one model to be a specific variant of another by applying model reductions based on linguistic similarity between activity descriptions in both business process models. An approach that describes similarity between processes based on the structural measure is proposed in [16]. The approach based on the theory of graph matching. Each BPEL process is translated into a graph. Two BPEL graphs are compared using the notion of edit distance which means that matching is done entirely on the syntax level where behavioral aspects are ignored. In [17] the authors discuss behavioral similarity of business process models. It is proposed to construct causality graph footprint vectors and to derive similarity as a cosine of the angle between two footprint vectors. The approach requires that similar activity models are exactly the same in compared process models, *i.e.*, have the same names, while our work integrates the similarity measurement of activity names and combine the structural and behavioral similarities of business processes.

Services discovery and recommendation. The efficiency problems of composite service are explored recently, and there are some useful relative research works. Evren Sirin et al. [18] presented an approach for semi-automatic services composition. The basic idea of their work was that the matching services at each step of the composition were filtered based on the contextual information and user's decisions. The matches for the parameters, between the service in current process and optional service, gave the result for service selection. Aliaksandr et al. [19] gave a method for improvement of services discovery based on history usage data, which could conduct corresponding rules by analyzing the service access history record. The services recommendation by Rosanna et al. [20] for services composition also reused the knowledge about past experience. Those experience data included non-functional attribute information such as location, and also used ontology for usage data learning and recommendation. The works mentioned above gave us a new usage pattern of services discovery, but they all concentrate on the atomic services and have nothing to do with the behavior of composite services.

8. Conclusion and future work

In this paper we proposed BestRec, a novel services recommendation approach to Web services composition based on the behavioral similarity of the developing composite service and existing services. The contributions of this paper include: (1) we define a NFA-based service model denoted by *NSM* to describe the composite service and its behavioral characteristics; (2) we give an algorithm to represent a composite service modeling with BPMN with *NSM*; (3) finally we evaluate the behavioral similarity of two *NSMs* through simulations, and the services recommendation mechanism is discussed.

Currently, BestRec has been used to recommend appropriate candidate services to developers. However, how to use the recommended services is highly dependent on developers' experience. Our future work will lead to further enhancing services composition efficiency by investigating the referenced edit operation set on transferring the developing service to the recommended services.

Acknowledgment: This work was supported by the National High Technology Research and Development Program of China (863 program) under grant 2007AA010301, 2006AA01A106 and 2009AA01Z419.

References

[1] Zhang, L.-J., J. Zhang, and H. Cai, Services Computing. 2007: Beijing : Tsinghua University Press.

[2] RosettaNet. Partner Interface Process, V2.05.00. RosettaNet Program Office, October 2008.

[3] Z. Shen, J. Su, Web services discovery based on behavior signatures. In Proceedings of IEEE SCC, 2005.

[4] D. Grigori, J.C. Corrales, and M. Bouzeghoub. Behavioral matchmaking for service retrieval. In Proc. of ICWS, 2006.

[5] White, S.A. (2008-01-17) Business Process Modeling Notation, V1.1. http://www.omg.org/spec/BPMN/1.1/PDF Volume

[6] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: A new approach to design and analysis of e-service composition. In Proceedings of 12th World Wide Web Conference (WWW), pages 403 - 410, May 2003.

[7] Aalst, W.v.d. and K.v. Hee, Workflow Management

Models, Methods, and Systems. 2002, Massachusetts London, England: The MIT Press Cambridge.
[8] J.C. Corrales, D. Grigori, and M. Bouzeghoub. Bpel processes matchmaking for service discovery. In Proc. CoopIS 2006, Lecture Notes in Computer Science 4275, pages 237.254. Springer, 2006.
[9] J. Mendling, B. Dongen, and W. Aalst. On the Degree of Behavioral Similarity between Business Process Models. In CEUR-Workshop, pages 39–58, 2007.
[10] R. Milner. Communication and Concurrency. Prentice-Hall, New York, 1989.
[11] Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S.M., Zave, P.: Matching and merging of statecharts specifications. In: ICSE. (2007) 54–64
[12] O. Sokolsky, S. Kannan, and I. Lee. Simulation-based graph similarity. In TACAS, pages 426–440, 2006.
[13] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of statecharts specifications. In ICSE ’07, pages 54–64,Washington, DC, USA, 2007. IEEE Computer Society.
[14] Günay, A., Yolum, P.: Structural and semantic similarity metrics for Web service matchmaking. In: EC-Web 2007. Volume 4655 of LNCS., Springer (2007) 129-138
[15] R. Lu and S. W. Sadiq. Managing Process Variants as an Information Resource. In Business Process Management, pages 426–431, 2006.
[16] J.C. Corrales, D. Grigori, and M. Bouzeghoub. Bpel processes matchmaking for service discovery. In Proc. CoopIS 2006, Lecture Notes in Computer Science 4275, pages 237.254. Springer, 2006.
[17] J. Mendling, B. Dongen, and W. Aalst. On the Degree of Behavioral Similarity between Business Process Models. In CEUR-Workshop, pages 39–58, 2007.
[18] Sirin E., Parsia B., Hendler J. Composition-driven filtering and selection of semantic web services. In American Association for Artificial Intelligence Spring Symposium on Semantic Web Services. 2004
[19] Birukou A., Blanzieri E., Andrea V.D., Giorigini P.,et, al. Improving Web Service Discovery with Usage Data[J]. IEEE SOFTWARE 2007, November:47-54
[20] Bova R., Paik H., Hassas S., Benbernou S, Boualem Benatallah. WS-Advisor: A Task Memory for Service Composition Frameworks [A]. Computer Communications and Networks [C]. 2007:535-540