

Structure Pattern Oriented Method for Service Composition

Minghui Wu^{1,2}, Canghong Jin¹, and Jing Ying^{1,2}

¹College of Computer Science, Zhejiang University, Hangzhou, P.R.China

²Department of Computer Science, Zhejiang University City College, Hangzhou, P.R.China

Email: {minghuiwu}@cs.zju.edu.cn

Abstract—Improving dynamic feature of service composition is a vital and challenging stage in web service application. With the increasing of service number, it is hard to balance the flexibility and verification very well in dynamic and complex environment. The previous methods compare two services by their own features. These methods are useful and concise to find substitutes for unavailable services, but they might be too strict to find a solution on some occasions. We propose a novel method called SPACE to build and estimate service composition. The method uses six composite patterns as basic elements to organize services and it also measures impact of service replacement on these patterns in stead of pair of single ones. By using SPACE, we can find non-optimal solution for service selection and re-planning in service composition.

Index Terms—web service, situation calculus, composite pattern

I. INTRODUCTION

Web service has become one of the most popular research fields recently. The process-based approach for web service composition has gained considerable momentum and standardizations. However, this service-centric approach can not run very well in a dynamic environment for its hard and pre-defined code description. In order to create dynamic composite systems, developers should not only define a suitable way to present web services which could well support environment adaptability, but also need to design a suitable composite approach which could easily measured. Currently, there are many approaches of web service composition. We categorize them into three different types: template-based composite [9] [11], interface-based composite [5] [7] and functionality-based composite [4]. Some of them are lack of adaptability and flexibility and others can not verify the business process very well.

Another big problem of dynamic service composition is how to reselect and re-plan service when original system is unavailable. Web service semantic is usually considered to measure similarity among different providers and approaches based on semantic is used to select web services. But nowadays semantic web service selection and replacement methods lack process information and hardly validate business process correctness. We also concern that evaluating the difference between original services and their substitutions alone and out of their context environment, which is very popular in industry and search area and

very precise and clear to distinguish the difference of two services, may be too strict to find a suitable one on some occasions. Isolated evaluation also might aggrandize or ignore the influence of whole business process caused by their dissimilarities. For example, in sequence structure, if the output of one service is changed, it may break up the business process. But if this service is in XOR structure, the whole business might only lost a part of its logic. We can infer that these differences are brought by the different structures they belong to.

According to the description above, there are two questions generated: one is how to describe service function and relationship formally. The other is how to measure the impact caused by service replacement. We try to find a proper solution to answer these questions. We assume that the process flow contains business useful information and it should be considered in service composition. Base on this assumption, we propose a method called Structure Process Analyze based Composition Environment (SPACE) to define, describe and evaluate service composition formally. SPACE method, generated from IBS and FBS, uses some process patterns as its basic units and expresses its business logic and constraints by situation calculus. SPACE architecture still uses IOPE to describe service function and defines internal constraint and external constraint to document relationship between atomic services. The influence of service replacement is according to these constraints. Finally, SPACE provides a formula to evaluate similarity (impact of service) of original and changed environment. Benefits of SPACE are evaluating web service composition by both semantic and its pattern, which could ensure the composite correctness of structure, synaptic and semantic.

This article is structured as follows. In Section 2, basic process structures are expressed and five web service replacement types are defined. In this section, we also give a useful algorithm to evaluate the similarity of a pair of services by their structures. Section 3 introduces the related work. Finally in Section 4, we present conclusions and future work.

II. SPACE ARCHITECTURE

SPACE approach defines features of six basic structure patterns with situation calculus notations and uses data stream to connect these patterns. We also define five different replacement rules and give each of them a weight and calculate impact by these values.

A. Structure Pattern of SPACE

In SPACE, business partners are considered as atomic services which only supply one service or function at one time. It is need to find a process language to connect these atomic services and present business logic. The situation calculus language (SC) [10] is a first-order logical language for representing dynamical changing worlds in which all of the changes are the direct result of named actions performed by atomic service. In situation calculus, situations are considered as a sequence of actions. Some notations include means of representing knowledge are used in our approach. You can find the detail information from article [10].

Atomic service $a(y)^s$ in our article is the one where a single Web-accessible computer program, sensor, or device is invoked by a request message, performs its task and perhaps produces a single response to the requester. For simply, we first use situation calculus to present atomic service and its constraints. Like traditional methods, we still use input, output, precondition, and effect (colloquially known as IOPEs) to present a service behavior and ignore service un-function attributes as cost, response time and so on. So the semantic meaning of IOPE could be described as follows:

Precondition: $Poss(a, s) \rightarrow Kref(\varphi_1, s) \wedge \dots \wedge Kref(\varphi_n, s)$

Input: $Poss(a, s) \rightarrow \pi_1 \wedge \pi_2 \wedge \dots \wedge \pi_n$, where π is the input data.

Output: $Poss(a, s) \wedge a$ or $do(a, s)$

Effect: $Poss(a, s) \wedge r_F^+(x, a, s) \rightarrow F(x, do(a, s))$ or

$Poss(a, s) \wedge r_F^-(x, a, s) \rightarrow \neg F(x, do(a, s))$

Giving the semantic meaning of atomic service is not necessary because the environment is more complex in real world. It needs more than one atomic services cooperated to complete a business process. But no matter how complicate the business flow is, it always can be divided into some smaller structures unless all of the services are atomic services. In Article [12], the author expresses six service composition patterns (CS): Sequential, AND split, XOR split (conditional), Loop, AND join (Merge) and XOR join (Trigger). Constraints between atomic services and structures are similar with user constraints, which contain interface type matching, semantic domain matching and other un-process logic elements. We define two types of constraints for SPACE: Internal constraint controls the interface between atomic services. External constraint controls the communication between structures. Three verifications could be achieved by basic patterns and their constraints.

The key questions are how to formally express these six patterns by atomic elements, and how to give the relationships of atomic services in a basic pattern. We give some definitions to answer these questions. Due to the limited of page, we only give two classic pattern definitions, Sequential pattern and XOR split pattern, and others are similar with these two.

Definition1 (Sequential Pattern) also can be called serial means a task is enabled after the completion of another task. Suppose there are two atomic services a_i and a_j ,

then the Sequential Pattern $CS_{seq}(a_i, a_j)$ constraints could be described by its IOPEs:

Precondition:

$Poss(a_i, s) \wedge Poss(a_j, do(a_i, s)) \rightarrow Kref(\varphi_1, s) \wedge \dots \wedge$

$Kref(\varphi_n, s) \wedge Kref(\varphi_1, do(a_i, s)) \wedge \dots \wedge Kref(\varphi_n, do(a_i, s))$

Input:

$Poss(a_i, s) \wedge Poss(a_j, do(a_i, s)) \Rightarrow Poss(a_i, s) \rightarrow$

$\pi_1 \wedge \pi_2 \wedge \dots \wedge \pi_n$

Output:

$Poss(a_j, do(a_i, s)) \wedge r(x, a_j, do(a_i, s)) \rightarrow Knows(y, do(a_j, do(a_i, s))) \wedge Kwhether(y, do(a_j, do(a_i, s))) \wedge Kref(y, do(a_j, do(a_i, s)))$

Effect:

$Effect : Poss(a_i, s) \wedge r_F^{+/-}(x, a_i, s) \wedge Poss(a_j, do(a_i, s)) \wedge r_F^{+/-}(x, a_j, do(a_i, s))$

Internal Constraint:

$Poss(a_j, do(a_i, s)) \rightarrow do(a_i, s)$

$Poss(a_0, s) \wedge r_F^{+/-}(x, a_0, s) \wedge Poss(a_i, do(a_0, s)) \wedge r_F^{+/-}(x, a_i, do(a_0, s))$

Internal constraint:

$Poss(a_1, do(a_0, s)) \rightarrow do(a_0, s) \wedge \dots \wedge Poss(a_n,$

$do(a_0, s)) \rightarrow do(a_0, s)$

Definition2 (XORsplit Pattern) also be called conditional routing or switch means process on a condition, one of services branches is chosen. The form of XORsplit is $CS_{XORS}(a_0, a_1, \dots, a_n, c_1, \dots, c_n)$, where a_0 is an initial service, a_i will be chosen when c_i condition is occurred. The IPOEs of XORsplit Pattern are:

Precondition:

$Poss(a_0, s) \wedge (Poss(a_i, do(a_0, s)) \wedge r_F^+(c_i, a_0, s) \wedge do(a_0, s))$

Input:

$Poss(a_0, s) \rightarrow \pi_1 \wedge \pi_2 \wedge \dots \wedge \pi_n$

Output:

$Poss(a_i, do(a_0, s)) \wedge r(x, a_i, do(a_0, s)) \wedge do(a_0, s) \rightarrow Knows(y, do(a_i, do(a_0, s))) \wedge Kwhether(y, do(a_i, do(a_0, s))) \wedge Kref(y, do(a_i, do(a_0, s)))$

Effect:

$Poss(a_0, s) \wedge r_F^{+/-}(x, a_0, s) \wedge (Poss(a_i, do(a_0, s)) \wedge r_F^{+/-}(x, a_i, do(a_0, s)) \wedge do(a_0, s))$

Internal constraint:

$do(a_0, s) \in \{Poss(a_1, do(a_0, s)), \dots, Poss(a_n, do(a_0, s))\}$

All the patterns are defined by IOPE of atomic services. We use pattern rather than atomic service as basic unit to organize business logic. With internal constraint and external constraint, business logic and service composition structure can be verified and maintained. Moreover, the impact of service replacer to business process does not only depend on change of single service but also depend on change of structure which this atomic service belongs to.

B. Evaluate the Impact of Service Replacer by Constraints

In above section, we use situation calculus to present six patterns which seem as basic elements to build business logic. According to formal structure express and two types of constraints, we propose a novel approach to evaluate the influence by atomic service replacement and select a suitable (maybe not optimal) service as substitute. As mentioned in Section 1, if there is no absolutely compatible candidate service could be chosen, the non-optimal substitute should maintain the structure of composite logic and limit the influence to other logic process.

There are many approaches to document the relationship of two single services. SPACE extends these methods and promotes them to fit pattern comparison. We use IOPE to describe the semantic of replacement type as well. In this paper, we assume input and output data are only complicate type, which contains basic data format and special domain they belong to, so we can use ontology to draw the hierarchical relationship and their semantic meanings. Now, five types of replacement are defined as follows, in which IC means internal constraint and EC means external constraint.

Definition3. Equivalent Replacement (EQU): service a_i and a_j are equal means

$$EQU(a_i, a_j) \equiv ICa_i = ICa_j \wedge ECa_i = ECa_j \Leftrightarrow$$

$$IOPEa_i \subseteq ICc_{s_i} \wedge IOPEc_{s_i} = IOPEc_{s_j}$$

EQU states both internal and external constraints of two services are the same.

Definition4. Include replacement (INC): service a_i and a_j are include means

$$INC(a_i, a_j) \equiv ICa_i \approx ICa_j \wedge ECa_i \subseteq ECa_j \Leftrightarrow$$

$$IOPEa_i \subseteq ICc_{s_i} \wedge (Ic_{s_i} \subseteq Ic_{s_j} \vee Oc_{s_i} \subseteq Oc_{s_j} \vee Pc_{s_i} \subseteq$$

$$Pc_{s_j} \vee Ec_{s_i} \subseteq Ec_{s_j})$$

INC states internal constraints of two services are compatible and at least one of external constraint of serviceTwo(e.g. Input is Man) is contained by the one of serviceOne(e.g. Input is person).

Definition5. Coverage replacement (COV): service a_i and a_j are coverage means

$$COV(a_i, a_j) \equiv ICa_i \approx ICa_j \wedge ECa_j \subseteq ECa_i \Leftrightarrow$$

$$IOPEa_i \subseteq ICc_{s_i} \wedge (Ic_{s_i} \supseteq Ic_{s_j} \vee Oc_{s_i} \supseteq Oc_{s_j} \vee Pc_{s_i} \supseteq Pc_{s_j} \vee Ec_{s_i} \supseteq Ec_{s_j})$$

Contrary to INC, COV means semantic of substitute can contain those of original one.

Definition6. Intersect replacement (INT): service a_i and a_j are intersect means

$$INT(a_i, a_j) \equiv ICa_i \approx ICa_j \wedge ECa_j \nabla ECa_i \Leftrightarrow$$

$$IOPEa_i \subseteq ICc_{s_i} \wedge (Ic_{s_i} \nabla Ic_{s_j} \vee Oc_{s_i} \nabla Oc_{s_j} \vee Pc_{s_i} \nabla Pc_{s_j} \vee Ec_{s_i} \nabla Ec_{s_j})$$

$$A \nabla B = (A \cap B) \neq \phi \wedge (A \setminus B \neq \phi)$$

INT states internal constraint of two services is compatible, and external constraints of services are only partially compatible. For example, output of service *buyFordCar* is INT with that of service *buyHondaCar*.

Definition7. Exclusion replacement (EXC): service a_i and a_j are exclude means

$$EXC(a_i, a_j) \equiv ICa_i \neq ICa_j \vee ECa_j \subseteq \neg ECa_i \Leftrightarrow$$

$$IOPEa_i \not\subseteq ICc_{s_i} \vee (Ic_{s_j} \subseteq \neg Ic_{s_i} \vee Oc_{s_j} \subseteq \neg Oc_{s_i} \vee Pc_{s_j} \subseteq \neg Pc_{s_i} \vee Ec_{s_j} \subseteq \neg Ec_{s_i}) \vee \perp$$

EXC states either internal or external constraint of two services is absolutely incompatible.

Until now, five replacement types are given. In actual world, one atomic service replacer might relate more than one replacement types. For example, there are two services *ServiceA* and *ServiceB*. Input, Output and Effect of *ServiceB* may be equivalent to those of *ServiceA*. But precondition and composite structure of *ServiceB* are totally different from those of *ServiceA*. The similarity value between *ServiceA* and *ServiceB* should combine all these facts.

The last part of SPACE is how to evaluate the impact of service replacement. The impact measure depends on the basic structures, constraints and replacement types defined above. The impact measure system is $\langle \mu, S, RT, f, TH \rangle$ where:

- μ is the weight value for each feature of service which is set by user's preference.
- S is the concept similarity of two services. It depends on external resources to store semantic information. But the types of interface of two services should be compatible.
- RT is the types of replacement. We set a value to each type for calculate. In order to keep the semantic consistency, the values of are set by following rule:
 $I = EQC > INC > COV > INT > EXC = 0$
- f is the function feature of service. It is consisted by input, output, precondition and effect.
- TH is the threshold value for service replacement. The candidate service wouldn't be chosen if its value can not exceed the TH . The value of TH setting should consider the value of μ .

Measuring the semantic similarity or relatedness between a pair of concepts is a complex task. There exist many approaches in AI area to research and improve the algorithm for measuring. Because estimating of semantic relatedness between words is not the key point in SPACE, we just choose a simple but resolute formula to calculate similar for S .

$$Sim(W_1, W_2) = \frac{\alpha}{Dis(W_1, W_2) + \alpha} \quad (1)$$

Where $Dis(W_1, W_2)$ is the distance of two services and α is the adjustable parameter whose value equals the words distance when their similarity is 0.5. In SPACE, one atomic service always belongs to more than one structure (e.g. Service WC in health check scenario is in both SEQ and ANDS structures), thus we need to

consider all the related structures when estimating the influence.

$$IM = \sum_{i \in CS} (\sum (\mu_f \cdot S_f \cdot RT_f) / M) / N \quad (2)$$

where f is set of IOPE, M is number of IOPE, CS is set of structures which atomic service belongs to and N is number of CS . From equation (1) and (2), two services are more similar and the influence is smaller if the values of these equations are higher. Value is zero states the substitute is completely incompatible in original environment.

III. RELATED WORK

There are various research activities in dynamic services composition and semantic web service. For instance, some approaches based on process description extend existing techniques like BPEL or OWL-S to present services composition. WS-BPEL, the most candidate standard for web services orchestration, provides some mechanisms to present long-running transactions and error handling. A formal description method which based on PI calculate and BPEL is used to present web services process [5]. Abstract state machine (ASM) defines operational semantics for BPEL and it can provide a comprehensive and robust formalization [6]. Article [7] establishes formal model of services according to FOCUS theory [8], and uses data stream to present architectures, structure and service behaviors. These approaches mentioned above do not cater for flexible and adaptive business collaborations because process should be pre-defined and can hardly be changed.

Other approaches propose to make web services composition dynamic and self-adapted. Business Collaboration Development Framework (BCDF) [4] creates different behaviors and different layers to express business collaboration. Different types of rules are defined to describe, constrain and control the operations and strategies of business. Article [9] introduces a meta-model which can evaluate parameter values at run time, so WS-flows flexibility can be improved. Mark Carman attempts to view service composition problem as a planning problem and uses document to describe service function and user goals [1]. This approach gives semantic relationship by using WordNet and chooses web services according with their interface type matching. Article [2] proposes the semantic relations by precondition and postcondition. In this paper, authors define four types of service relationships and four types of service match. These articles are very useful for our SPACE architecture and our proposal is based on some of their ideas and methods.

V. CONCLUSION

Current web service composition methods either need pre-define process or just connect by service interface, the disadvantages of these approaches are preclude the business dynamics or too flexible to verify service

correctness. The challenge is how to balance the flexibility and correctness of service composition.

In this paper, we propose a novel approach called SPACE architecture which uses situation calculus to express basic patterns of business process and defines the internal and external constraints. Moreover, SPACE classifies the service replacement categories based on these structures and constraints. A similar estimating formula is also given to measure the impact of service replacement. Based on these features, SPACE can provide more “fairly” and precisely solution to choose a substitute for unavailable service.

Work for future research will foremost focus on incorporation of structure, semantic and QoS. We are also working on building the platform of SPACE.

ACKNOWLEDGMENT

This work was supported in part by the National High-Tech Foundation (863), China (Grant No.2007AA01Z187).

REFERENCES

- [1] Mark Carman, Luciano Serafini, Paolo Traverso, Web Service Composition as Planning, ICAPS, 2003.
- [2] Lin Lin, Arpinar I.Budak, Discovering Semantic Relations between Web Services Using Their Pre and Post-Conditions, ICWS apos, 2006.
- [3] Shankar R. Ponnekanti, Armando Fox, SWORD:A Developer Toolkit for Web Service Composition, The 11th International WWW2002.
- [4] Bart Orriens, Jian Yang, Mike Papazoglou, A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaboration, ICSOC 2005, LNCS 3826, pp.61-72.
- [5] Roberto Lucchi, Manuel Mazzara, A pi-calculus based semantics for WS-BPEL, The Journal of Logic and Algebraic Programming 70(2007) 96-118.
- [6] Roozbeh Frarhbod, Uwe Classer, Mona Vajiholahi, A Formal Semantics for the Business Process Execution Language for Web Services.
- [7] Manfred Broy, IngolfH Kruger, Michael Meisinger, A Formal Model of Services, ACM Transactions Software Engineering and Methodology, Vol.16, No.1,Article5, February,2007.
- [8] Broy M, Stolen K, Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement, Springer Verlag, New York, 2001.
- [9] Dimka Karastoyanova, Frank Leymann, Alejandro Buchmann, An Approach to Parameterizing Web Service Flows, ICSOC 2005, LNCS 3826, pp.533-538, 2005.
- [10] Srinu Narayanan, Sheila Mcilraith, Analysis and Simulation of Web services, Computer Networks: The International journal of Computer and Telecommunications Networking, 2003, 42 (5): 675-693.
- [11] Uwe Zdun, Carsten Hentrich, Schahram dustdar, Modeling Process-Driven and Service-Oriented Architecture Using Patterns and Pattern Primitives, ACM Transactions on the Web, Vol.1, No.3, September, 2007.
- [12] Tao Yu, Yue Zhang, Kwei Jaylin, Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints, ACM Transactions on the Web, Vol.1, No1. May, 2007