A Structure-based Approach for Dynamic Services Composition

Canghong Jin
College of Computer Science, Zhejiang University, Hangzhou, P.R.China
Email:{canghong.jin}@gmail.com

Minghui Wu* and Jing Ying
College of Computer Science, Zhejiang Univerisity, Hangzhou, P.R.China
Department of Computer Science, Zhejiang University City College, Hangzhou, P.R.China
Corresponding author email:{minghuiwu}@cs.zju.edu.cn

***Abstract*—Web service composition needs to increase the dynamic feature of adapting complex and unstable business environment. Thus the service selection and evaluation are very important for service composition. Although previous studies have stated some approaches to support dynamic composition, they are unable to balance the flexibility and verification very well in dynamic environment. Moreover, most service selection methods only depend on the similarity of a pair of single services. These methods are useful and concise to find substitutes for unavailable services, but sometimes they might be too strict to find a solution. To overcome these problems, we propose a novel approach named SPACE architecture with six basic structures, and define the service composition based on situation calculus language. SPACE estimates the similarity by the basic structures and constraints rather than the features of single service. It can help us to find non-optimal but acceptable substitutes and guarantee the verifications of composition. Finally, as a case study we consider a health care scenario to demonstrate our approach.**

***Index Terms*—web services composition, composite structure, SPACE architecture, situation calculus**

I. INTRODUCTION

Web services are considered to be the web's next revolution and the future of e-business. Web services are "self-contained, self-describing, modular applications that can be published, located, and invoke across the Web" [2]. It 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] [6] [7] and functionality-based composite [4] [10]. We give the more detail information about these compositions. Template based system (TBS) composes an application from a given service template, and is well structured and validated easily. TBS limits adaptability. One special template can only suit for one particular environment. Interface based System (IBS) uses inputs and outputs information to connect different components, so it has higher adaptability, but the correctness of service function can not be guaranteed. Functionality based System (FBS) is based on IBS and adds some logic elements such as pre-condition and post-condition to data flow. FBS is well organized in special domain and can run well on one domain, but maybe not fit for others. Developers need to create a logic rule for each domain, which limit the reusability of FBS.

Another big problem of dynamic service composition is how to reselect and re-plan service when original system is unavailable. Web service has its function attribute and non-function attribute. The former defines what the service can do and the later, considered as QoS, describes the quality of service. Re-selecting and re-planning of service composition are always according to these attributes. In this paper, we only consider function attribute of service and ignore the impact of non-function attribute though it is also very important for service composition. 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 are 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 in some occasions. Isolated evaluation also might aggrandize or ignore the influence of whole business process caused by their dissimilarities. We will give two simple scenarios to express this phenomenon.

In general, a web service operation is specified by its Input message, Output message, Precondition and Effect.

doi:10.4304/jsw.4.8.891-898

We view service operations as the operators available to the composite system. For example, a manager wants to travel from Hang Zhou to New York to attend a meeting. Assume there is no direct flight line between these cities. He needs to book air ticket and train ticket by his credit card. In this scenario one, we define S_2 as *bookTrainTicket* and S_3 as *bookAirTicket*. The input message of S_2 is *CreditCardNumber* and *TrainNumber*. The output message of S_2 is *TrainTicket*. The precondition of this service is *checkCreditCard* which means before *bookTrainTicket* the system should make sure the manager's credit card is available. Other Services and attributes are not the key points in this example. We could assume S_1 is *checkTravelDate* and S_4 is *bookHotel*. As all the atomic services are defined, we combine them together to build business logic (see Figure 1). We call this structure is AND structure for S_2 and S_3, which states S_2 and S_3 should be finished without exception before S_4 begin to run. Detail explanation for AND structure will be given in Section 3. For some unpredicted reasons, service S_2 is unavailable and S_5 is considered as a backup service. Unfortunately, S_5 is not complete the same as S_2. It will not do *checkCreditCard* (precondition constraint) before *bookTrainTicket*. It seems S_5 is unsuitable as a replacer if *checkCreditCard* is very important for business logic. But S_5 could run well with no problem in scenario one for AND structure it belongs to (see Figure3). Because S_2 will do the precondition *checkCreditCard* for S_5 if they use the same credit card, this composition structure still maintains the original business logic after service replacement. However, in scenario two, we suppose this manager will travel from Hangzhou to Beijing and he can choose either by train or by flight. The features of web services are the same as those in scenario one. Business logic is organized as Figure 2, which presented by XOR structure. If S_2 is replaced by S_5 (see Figure 4) in XOR structure, this structure can not keep the semantic of composite services and the business logic is changed. That is to say, if manager choose service *bookTrainTicket*, system can not supply the service *checkCreditCard* for business user.

Although the scenarios we presented are very simple and might conflict with real life (e.g., why user use the same credit card in scenario one), it expresses the result clearly: Even putting the same service into different structures may generate different impact to original environment and sometimes these differences are undetected if only comparing sole services. Although traditional methods of comparing features of single service are very useful and effective, sometimes they might discard those non-optimal services which can run in original system as well. One goal of this paper is to find out these non-optimal services as substitutes for service composition.

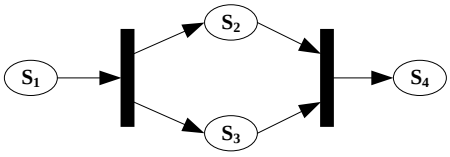

Figure 1. service composition in AND structure

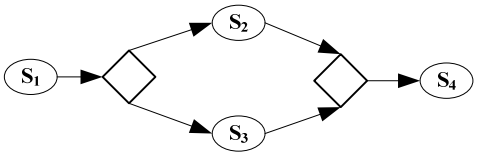

Figure 2. service composition in XOR structure

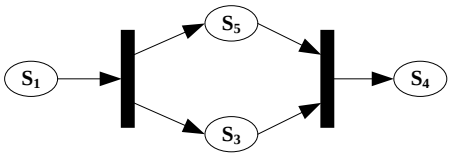

Figure 3. service replacement in AND structure

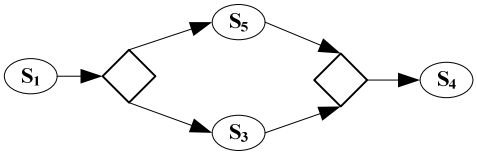

Figure 4. service replacement in XOR structure

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 these questions. We find the process flow contains business useful information. So it should be considered during web services re-selecting and re-planning. 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 structure as its basic unit 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. Innovative features of SPACE are evaluating web service composition by both semantic and its structure, which could ensure the composite correctness of structure, synaptic and semantic.

This article is structured as follows. In Section 2 introduces the related work. Basic process structures are expressed and five web service replacement types are defined in Section 3. In this section, we also give a useful algorithm to evaluate the similarity of a pair of services by their structures. In Section 4 demonstrates a health care scenario to present our SPACE approach and finally in Section 5, we present conclusions and future work.

II. 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 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. Dynamic Service-Oriented Architecture (DySOA) [10] extends service-centric applications, it uses four components to monitor, analysis, evaluate and configure web services. So DySOA can make services adaptable at runtime. Mark Carman attempts to view service composition problem as a planning problem and uses document to describe service function and user goals [17]. This approach gives semantic relationship by using WordNet and chooses web services according with their interface type matching. Article [18] proposes the semantic relations by precondition and postcondition. In this paper, authors define four types of service relationships and four types of service match. Authors in [1] use situation calculus to document service composition and user constraints and propose exception patterns to deduce the planning procedure. These articles are very useful for our SPACE architecture and our proposal is based on some of their ideas and methods.

III. SPACE ARCHITECTURE

SPACE architecture in this section splits the verification in model checking into three different kinds: syntactic verification, semantic verification and structural verification. SPACE approach defines features of six basic structures with situation calculus notations and uses data stream to connect these structures. We also define five different replacement rules and give each of them a weight and calculate impact by these values.

A. Concept and structure of SPACE

Process description methods, such as BPEL or e-flow, introduce a state model of web services interacting by exchanging sequences of data between business partners. 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) [13] 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. SC can create an optimized plan for various domains, so it can be used to express dynamical world [1]. In situation calculus, situations are considered as a sequence of actions. Some notations include means of representing knowledge are used in our approach. These notations extend the basic meanings in situation calculus language to be adapted in SPACE. Table 1 shows these notations and their related semantic meanings.

TABLE I
SITUATION CALCULUS NOTATIONS IN SPACE

$a(y)^s$	atomic service a in situation s
$F(x_1, x_2, ..., s)$	fluent from situation to situation, x_i is argument,
$do(a,s)$	result of performing a in situation s
$Poss(a,s)$	atomic service a is possible to perform in situation s
$K(s,s')$	accessibility relation between situation s and s'
$Knows(\Phi,s)$	Φ is available in situation s
$Kwhether(\Phi,s)$	true value of Φ is available in situation s
$Kref(\Phi,s)$	function value of Φ is available in situation s

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 [15]. 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:

$Preconditi\,on: Poss(a,s) \rightarrow Kref(\varphi_1,s) \wedge ... \wedge Kref(\varphi_n,s)$

$Input: Poss(a,s) \rightarrow \pi_1 \wedge \pi_2 \wedge ... \wedge \pi_n$, 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 structures (CS): Sequential, AND split, XOR split (conditional), Loop, AND join (Merge) and XOR join (Trigger), see Figure5. 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 structures and their constraints.

a) Syntactic verification which verifies the model is in conformance with the grammar of the language. This verification can be guaranteed by both internal and external constraints.
b) Semantic verification which verify whether the model is in conformance with the business process goals. In this process, the contacts in structure are encapsulated and seem transparent to outside structures. That is to say, the structures encapsulate some sub-business logic and make process present more concisely. We also use external constraint to maintain semantic compatible.
c) Structural verification which is used to verify that the model will not lead to erroneous execution. In SPACE, internal constraint can verify the interface between atomic services and make the basic structures stable.

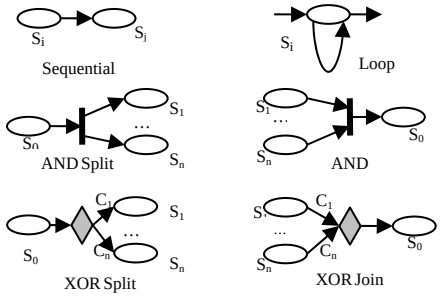

Figure 5. six composition structures

The key questions are how to formally express these six structures by atomic elements, which means how to express external constraint by internal constraint, and how to give the relationships of atomic services in a basic structure. We give the definition 1 to definition 6 to describe these questions.

Definition1 *(Sequential Structure)* 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 Structure $CS_{seq}(a_i, a_j)$ constraints could be described by its IOPEs:

Precondition:

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

Input:

$$Poss(a_i,s) \wedge Poss(a_j,do(a_i,s)) \Rightarrow Poss(a_i,s) \rightarrow \pi_1 \wedge \pi_2 \wedge ... \wedge \pi_n$$

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:

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

Internal Constraint:

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

Definition2 *(ANDsplit Structure)* also be called parallel split or fork means a single service splits into multiple services which can be executed in parallel. The form of ANDsplit is $CS_{ANDS}(a_0,a_1,...,a_n)$ where a_0 is an initial service and rest a_i are split services. The IPOEs of ANDsplit Structure are:

Preconditon:

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

Input:

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

Output:

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

Effect:

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

Internal constraint:

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

Definition3 *(XORsplit Structure)* also be called conditional routing or switch means process on a condition, one of services branches is chosen. The form of XORsplit is $CS_{XORS}(a_0,a_1,...,a_n,c_1,...,c_n)$, where a_0 is an initial service, a_i will be chosen when c_i condition is occurred. The IPOEs of XORsplit Structure 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:

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

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:

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

Definition4 *(Loop Structure)* means single service will be repeated until the condition is met. The form of Loop Structure is $CS_{Loop}(a,\lambda)$, where a is a service, λ is a condition. The IPOEs of Loop Structure are:

Precondition:

$$Poss(a,s) \wedge Poss(a,do(a,s)) \wedge Poss(a,do(a,do(a,s))) \wedge ...$$

Input:

$$Poss(a,s) \rightarrow \pi_1 \wedge \pi_2 \wedge ... \pi_n$$

Output:

$Poss(a,s) \rightarrow Knows(y,do(a,s)) \wedge Kwhether(y,do(a,s)) \wedge Kref(y,do(a,s))$

Effect:

$(Poss(a,s) \wedge r_F^+(x,a,s) \wedge \lambda) \vee ((Poss(a,do(a,s)) \wedge r_F^+(x,a,do(a,s)) \wedge \lambda)) \vee ...$

Internal constraint:

$(Poss(a,do(a,s)) \rightarrow do(a,s)) \wedge (Poss(a,do(a,do(a,s))) \rightarrow do(a,do(a,s))) \wedge ...$

Definition5 *(ANDjoin Structure)* also be called rendezvous or synchronizer means multiple parallel services converge into one single service. The form of ANDjoin Structure is $CS_{ANDJ}(a_0,a_1,...,a_n)$ where a_0 is a converged service and rest a_i s are initial services. The IPOEs of ANDjoin Structure are:

Precondition:

$Poss(a_1,s) \wedge Poss(a_2,s) \wedge ... \wedge Poss(a_n,s) \rightarrow Kref(a_1,s) \wedge ... \wedge Kref(a_n,s)$

Input:

$Poss(a_1,s) \wedge ... \wedge Poss(a_n,s) \rightarrow \pi_1 \wedge \pi_2 \wedge ... \pi_n$

Output:

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

Effect:

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

Internal constraint:

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

Definition6 *(XORjoin Structure)* also be called asynchronous join or merge means two or more alternative branches merge to one service. The form of XORjoin Structure is $CS_{XORJ}(a_0,a_1,...,a_n,c_1,c_2,...,c_n)$, if condition c_i is triggered then a_i will be merged to a_0. The IPOEs of XORjoin Structure 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:

$Poss(a_1,s) \wedge Poss(a_2,s) \wedge ... \wedge Poss(a_n,s) \rightarrow \pi_1 \wedge \pi_2 \wedge ... \pi_n$

Output:

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

Effect:

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

Internal constraint:

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

All the structures are defined by IOPE of atomic services. We can use structure 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 structures 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.

a) Make the structure as complete as possible means the new candidate service should not change the structure which can be guaranteed by its internal constraint. If basic structure is changed, the correctness of syntactic and structure verification would be broken up.
b) Make the service semantic as close as possible means the new candidate service should try to keep the semantics the more the better outside the basic structure. This goal can be guaranteed by service external constraint.

There are many approaches to document the relationship of two single services. SPACE extends these methods and promotes them to fit structure comparison. We use IOPE to describe the semantic of replacement type as well. Five categories are defined for our SPACE architecture. The input and output variables can be either primitive type or complicate type. In this paper, we assume them are only complicate type, which contains the basic data format and special domain they belong to, so we can use ontology to draw the hierarchical relationship and their semantic meanings. If we say the input data of service A is equal with that of service B, it means not only their data format types are compatible but also their object domains are alike equally. Now, five types of replacement are defined as follows, in which IC means internal constraint and EC means external constraint.

Definition7. *Equivalent Replacement (EQU)*: service a_i and a_j are equal means

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

$IOPEa_i \subseteq ICcs_i \wedge IOPEcs_i = IOPEcs_j$

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

Definition8. *Include replacement (INC)*: service a_i and a_j are include means

$INC(a_i,a_j) \equiv ICa_i \approx ICa_j \wedge \mathrm{E}Ca_i \subset \mathrm{E}Ca_j \Leftrightarrow$

$IOPEa_i \subseteq ICcs_i \wedge (Ics_i \subset Ics_j \vee Ocs_i \subset Ocs_j \vee Pcs_i \subset Pcs_j \vee Ecs_i \subset Ecs_j)$

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)*.

Definition9. *Coverage replacement (COV)*: service a_i and a_j are coverage means

$$COV(a_i, a_j) \equiv ICa_i \approx ICa_j \wedge ECa_j \subset ECa_i \Leftrightarrow$$
$$IOPEa_i \subseteq ICcs_i \wedge (Ics_i \supset Ics_j \vee Ocs_i \supset Ocs_j \vee Pcs_i \supset Pcs_j \vee Ecs_i \supset Ecs_j)$$

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

Definition10. *Intersect replacement (INT)*: service a_i and a_j are intersect means

$$INT(a_i, a_j) \equiv ICa_i \approx ICa_j \wedge ECa_j \nabla ECa_i \Leftrightarrow$$
$$IOPEa_i \subseteq ICcs_i \wedge (Ics_i \nabla Ics_j \vee Ocs_i \nabla Ocs_j \vee Pcs_i \nabla Pcs_j \vee Ecs_i \nabla Ecs_j)$$
$$A \nabla B = (A \cap B) \neq \phi \wedge (A \setminus B \neq \phi)$$

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*.

Definition11.*Exclusion replacement (EXC)*: service a_i and a_j are exclude means

$$EXC(a_i, a_j) \equiv ICa_i \neq ICa_j \vee \mathrm{E}Ca_j \subset \neg ECa_i \Leftrightarrow$$
$$IOPEa_i \not\subset ICcs_i \vee (Ics_j \subset \neg Ics_i \vee Ocs_j \subset \neg Ocs_i \vee Pcs_j \subset \neg Pcs_i \vee Ecs_j \subset \neg Ecs_i) \vee \perp$$

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 different replacement types. We evaluate similarity of two services not only by their input, output, precondition and effect but also by the basic structure they belong to. The reason has been given before. 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:*
  1= 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, as *lch*(Leacock & Chodorow 1998), *wup* (Wu & Palmer 1994) and *jcn* (Jiang & Conrath 1997) [19]. 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.

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

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.

IV. CASE STUDY: SIMPLE HEALTH CHECK

In order to demonstrate SPACE is applicable and useful, we use it to organize web services in a health care scenario which is also introduced in article [14]. This scenario is very simple and clear. It describes a patient goes to do health check and comprises sub scenarios as follows: firstly he does blood pressure and gains a warning level which based on blood pressure. Then according to the level of warning, both *MD* (medical department) and *ED* (emergency department) will assign their nurses or doctors to help the patient. There are six different web services in this scenario: service *BPC* (blood pressure check) returns the *BP* (Blood pressure) of a patient who has an identify *PID* (patientID) and an *ADDR(X)* (address of device X); service *SA* (staff assignment) return the supervisor (person) or a physician of an *ORG* (organization); service *WC* (warning classification) returns a *WL* (level of warning) and given *BP*(Blood pressure); service *AED* (assign emergency department) returns the *ED* and is given a level of warning; service *AO* (assign organization) returns the Organization and is given a level of warning. The process of this health care scenario, related web services and CS (composite structure) are presented in Figure 6.

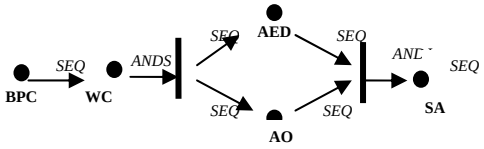

Figure 6. health care scenario composited by atomic services

The candidate services of *servceA* are named as *serviceA'*, *serviceA''*. For example, *BPC'* is considered to replace *BPC*. Services are described on their IOPEs which are showed in table2 and the composition structures these web services belong to are also showed in the same table.

TABLE II
CONSTRAINTS OF INITIAL SERVICE AND THEIR REPLACE SERVICE

Service	*Input*	*Output*	*Precondition*	*Effect*	*CS*
BPC	*PID, ADDR(BP)*	*BP*	*Patient*	*Log*	*SEQ*
SA	*ORG*	*Person*	*NULL*	*NULL*	*SEQ*
WC	*BP*	*WL*	*BPC*	*NULL*	*SEQ, ANDS*
AED	*WL*	*ED*	*WC*	*NULL*	*ANDS,ANDJ*
AO	*WL*	*MD*	*WC*	*NULL*	*ANDS, ANDJ*
BPC'	*PID, ADDR(BP)*	*BP*	*Person*	*Log*	
NT(Neural Test)	*PID, ADDR(X-ray)*	*Chest X-ray*	*Patient*	*Log*	
SA'	*MD*	*DOC*	*NULL*	*NULL*	
SA''	*Kennel*	*Dog*	*NULL*	*NULL*	
WC'	*BP*	*constantWL*	*NULL*	*NULL*	
AO'	*WL*	*ORG*	*WC*	*Log*	

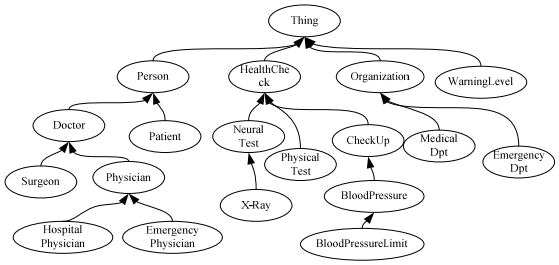

Figure 7 an e-healthcare taxonomic relationships or classes

As we mentioned above, we need external resources, like ontological language, to store information about IOPE. Figure7 shows the ontology taxonomic relationships in health care scenario. These relationships will be used to categorize replacement types. In this scenario, we suppose Effect is more important than IOP and the distance of two words is the length in hierarchy tree. Other parameters for influence measure are set in Table 3. Table4 gives the types of replacement by SPACE and similarity value of a pair of services.

TABLE III
PARAMETER VALUES FOR INFLUENCE MEASURE

parameters	*value*
α	*1*
μ *-Effect*	*2*
μ *-IOP*	*1*
EQC,INC,COV,INT,EXC	*1, 0.8, 0.5, 0.3, 0*

TABLE IV
TYPES OF SERVICE REPLACEMENT AND SERVICE SIMILARITY

	External Constraint				*Internal Constraint*	*Similarity*
	Input	*Output*	*Precondition*	*Effect*		
BPC, BPC'	EQU	EQU	INC	EXC	compatible	0.85
BPC, NT	INT	INT	EQU	EQU	compatible	0.78
SA, SA'	COV	COV	EQU	EQU	compatible	0.875
SA, SA''	EXC	EXC	EQU	EQU	incompatible	0
WC, WC'	INC( )	INC	EQU	EQU	compatible	0
AO, AO'	EQU	COV	EQU	EQU	compatible	1.06

From table 4, we find that only *AO'* could be chosen if user set value of threshold is one. Actually, set threshold process is very complex. It depends on the various parameters of formulas and user's preference. For service *BPC*, we can infer that *BPC'* is better than *NT*. There are two candidate services, *SA''* and *WC'*, are completely unacceptable in our measure. *SA''* uses *Kennel* as its input and *Dog* as its output which are completely incompatible with original IO parameters according to Figure 7. So its value should be zero. For *WC'*, although it only change a little about output, the influence is not so small as it seems. If we ignore the structure, the relatedness of output of *WC* and *WC'* is *INC*. It is correct in *SEQ* structure as well. However, *WC* is also in the structure *ANDS*, SPACE definition 2 states output of *WC* should contain both input of *AED* and *AO*. In our scenario, output of WC' is constant and it can not cover all the input situation thus it conflict with internal constraint. The structure *ANDS* will be changed to *SEQ* by the impact of *WC'*. So the similarity of *WC* and *WC'* should be zero for violating the structure verification.

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 structures of business process and defines the internal and external constraints. Business logical process could be divided into atomic services and re-combined with these structures. 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. SPACE would support service composition and selection by both service function attribute and service non function attribute. This states SPACE can use structure function selection to choose more candidate services, then SPACE can double choose these services by their non function attributes. Moreover, six structures we mentioned in this paper are only basic control flow elements, we need to extend them to meet more complex composition

environment. 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] Ken Nariai, Incheon Paik, Mitsuteru Shinozawa, Planning and Composition of Web Services with Dynamic Constraints Using Situation Calculus, CIT, 2005.
[2] D.T Tesemetizis, I.G. Roussaki, I.V. Papaionnou, M.E. Anagnostou, QoS awareness support in Web-Service semantics, AICT/ICIW 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 Parameterzing Web Service Flows, ICSOC 2005, LNCS 3826, pp.533-538, 2005.
[10] Johanneke Siljee, Ivor Bosloper, Jos Nijhuis, Dieter Hammer, DySOA: Making Service Systems Self-adaptive, ICSOC 2005, LNCS 3826, pp.255-268, 2005.
[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.
[13] Srini Narayanan, Sheila Mcilraith, Analysis and Simulation of Web services, Computer Networks: The International journal of Computer and Telecommunications Networking, 2003, 42 (5): 675~693.
[14] Freddy Lecue, Alain Leger, Semantic Web Service Composition Based on a Closed World Assumption, ECOWS, 2006.
[15] OWL-S: Semantic Markup for Web Services, w3c Member Submission 22 Nobember, 2004.
[16] Nicholas Gibbins, Stephen Harris, Nigel Shadbolt, Agent-based semantic Web services. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2004, 1(1): 141-154.
[17] Mark Carman, Luciano Serafini, Paolo Traverso, Web Service Composition as Planning, ICAPS, 2003.
[18] Lin Lin, Arpinar I.Budak, Discovering Semantic Relations between Web Services Using Their Pre and Post-Conditions, ICWS apos, 2006.
[19] Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi, WordNet::Similarity-Measuring the Relatedness of Cocepts, AAAI, 2004.

Canghong Jin, born in 1982, Master of Engineering Science. His research interests include service oriented software engineering and aspect-oriented software development.

Minghui Wu, born in 1976, Ph.D. candidate, Associate Professor. His research interests include software engineering, artificial intelligence and internet application.

Jing Ying born in 1971, Ph.D., Professor. His research interests include software engineering and software automation