2013 International Symposium on Theoretical Aspects of Software Engineering

Towards Conformance Testing of Choreography Based on Scenario

Hongli Yang,Kai Ma,Cheng Deng,Husheng Liao
College of Computer Science
Beijing University of Technology
Beijing, China
{yhl,mk,dc,liaohs}@bjut.edu.cn

Jun Yan,Jian Zhang
Laborary of Computer Science
Institute of Software
Chinese Academy of Sciences
{yj,zj}@ios.ac.cn

Abstract

Web service choreography specifies the interaction among multiple particpant, aiming to achieve common business goals. An issue is to check for the conformance of the implementation with reference to the choreography specification. To achieve that, we seek to develop software tools and a methodology to enable conformance testing of choreography. In this paper, we present our first step in that direction. Particularly, we reduce choreography scenario in order to obtain effective testing scenarios, which will greatly decrease the cost of testing. Concretely, based on XML Schema type definition of a choreography scenario, we partition XML Schema type into subtypes, which will be transformed into the input model of combinatorial tool Cascade for generating a set of combinations of variable values. The output of Cascade will be transformed to generate reduced scenarios for testing. Moreover, a purchase order choreography example is presented to demonstrate the reduction process of choreography scenarios, and a tool has been developed for supporting automatic implementation of the testing scenarios reduction.

Keywords: XML Schema; Web Service Choreography; Scenario; Combinatorial Testing

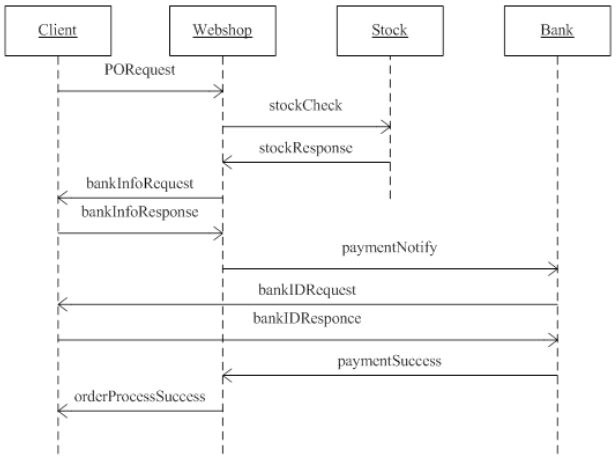

Fig. 1. A Successful Purchase Scenario

1. Introduction

A choreography specification identifies allowable ordering of message exchanges in a distributed system, and has been considered a scalable and decentralized solution for composing Web services. Choreography languages enable specification of such interactions. WS-CDL [1] is a typical example of the languages for describing global models of service interactions.

Choreography conformance problem is identifying if a set of given services adhere to a given choreography specification. It has been an important research area in service oriented computing in the last several years. There are some researches which present approaches for verifying the conformance [2], [3]. However, these works do not consider the data (the value of messages). They cannot be used to test Web service composition because they do not support data variable.

Since choreography describes the observable interactions among Web services, we consider conformance testing using the gray-box approach. In this approach, a composite of Web services can be specified by a WS-CDL language. An implementation of the composite Web service is then tested without any information of its internal structure. Test cases are generated only from WS-CDL specification.

The core problem we study in this paper focuses on the reduction of testing scenarios. Since a choreography scenario represents sequence of interactions among participants, we can use a choreography scenario to generate a set of testing scenarios, each of them is a possible execution paths that the composite Web services can follow. A choreography scenario involves not only a sequence of interactions, but also variable type declarations, which are defined in a referred XML Schema document. Due to the indicators such as $choice$, $minOccurce$ and $maxOccurce$, used in XML Schema type definition, a choreography scenario may be too complex to be used directly as a testing scenario. We need to reduce a choreography scenario based on XML Schema type definition.

In order to get effective testing scenarios, we will use combination approaches, which has been applied to the testing procedures [4]. Using combinatorial methods for testing can make testing more effective at an overall lower cost. This paper will use combinatorial tool $Cascade$[5] to reduce the testing scenarios.

Supported by open foundation of State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences (No. SYSKF1008), and Subject and Postgraduate Education Construction Project of Beijing Municipal Commission of Education.

978-0-7695-5053-4/13 $26.00 © 2013 IEEE
DOI 10.1109/TASE.2013.23

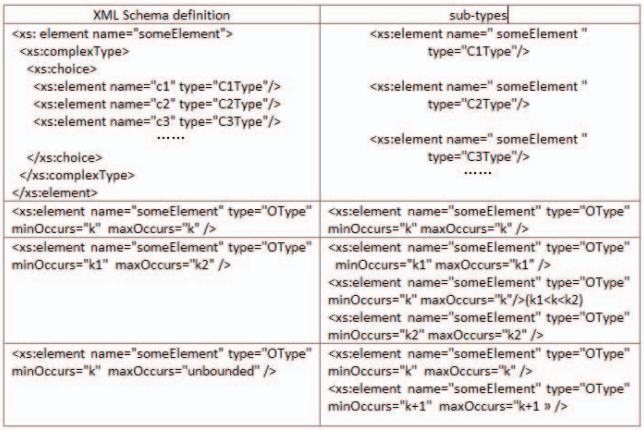

Fig. 2. XML Schema Type Partition Rules

2. A Purchase Order Choreography Example

This section presents a purchase order choreography example, which will be used through the paper for demonstrating the concepts and approaches. Figure 1 presents a UML sequence diagram for describing a successful purchasement scenario obtained from the example choreography.

A choreography describes a set of required scenarios, which can be extracted from choreography description. A choreography scenario describes a sequence of interactions among multiple participants. It defines expected behavior for all participants. Following is the formal definition of a choreography scenario.

Definition A choreography scenario is a 4-tuples $\langle R, I, V, A \rangle$, where R is a finite set of role declarations; I is a finite set of information type, which is defined in an external XML Schema document; V is a finite set of variable declarations, which map each variable with its type; A is a sequence of interactions.

3. XML Schema-based Type Partition

In this section, we will introduce type partition based on XML Schema definition. Since the type information of a scenario is defined in an XML Schema document, we know that XML Schema indicators can result in diversity of variable type. This section will present partition rules of XML Schema type. Particularly, we pay attention to three indicators: $choice, minOccurs$ and $maxOccurs$.

Figure 2 shows the rules of type partition based on the three indicators. Concretely, if $choice$ appears in a type definition, each choice branch needs to be extracted as a new element. If $minOccurs$ and $maxOccurs$ appear and take same value k, then the type definition keeps unchanged. If $minOccurs$ takes value k_1, and $maxOccurs$ takes value k_2 $(k_1 \neq k_2)$, we partition the type into three kinds of subtypes: (1) both $minOccurs$ and $maxOccurs$ take value k_1; (2) both $minOccurs$ and $maxOccurs$ take value k_2; (3) both $minOccurs$ and $maxOccurs$ take value k $(k_1 < k < k_2)$. If $minOccurs$ takes value k and $maxOccurs$ takes $unbounded$ value in a type definition, we simply partition the type into two subtypes: (1) both $minOccurs$ and $maxOccurs$ take value k; (2) both $minOccurs$ and $maxOccurs$ take value $k+1$. Here we assume that if a web service can correctly deal with these two kinds of subtypes, in the majority situation, it can also deal with other subtypes like $k+n$.

After partition, we can use subtypes of XML Schema to generate test data. Note that the derived instance based on them are still valid because they conform to the original XML Schema.

Following is the $purchaseOrder$ type definition in XML Schema document of the example choreography.

```
<xs: element name="purchaseOrder">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="goods"
    minOccurs="1" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="goodName"
           type="xs:string"/>
       <xs:element name="goodNum"
           type="xs:int" />
       <xs:element name="price" type="xs:int" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="coupon"
    minOccurs="0" maxOccurs="1" >
     <xs:ComplexType>
       <xs:choice>
       <xs:element name="discount"
           type="xs:decimal"/>
       <xs:element name="cashback"
            type="xs:string"
            minOccurs="0" maxOccurs="2"/>
      </xs:chioce>
     </xs:ComplexType>
    </xs:element>
    <xs:element name="address"
       type="xs:string" />
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

Based on the partition rules in figure 2, we can partition type $goods$ into two subtypes whose occurrence times is partitioned as 1 and 2. Similarly, we can partition element $coupon$ and $cashback$.

Since there are multiple variables in a choreography scenario, the variable types are defined in an XML Schema document, whose indicators may cause multiple values of elements in the type definition. The combination of multiple values of multiple elements will cause a big set of scenarios, which will increase the cost of testing. In order to resolve this issue, we will use combinatorial approach presented in section 4 to reduce the scenarios.

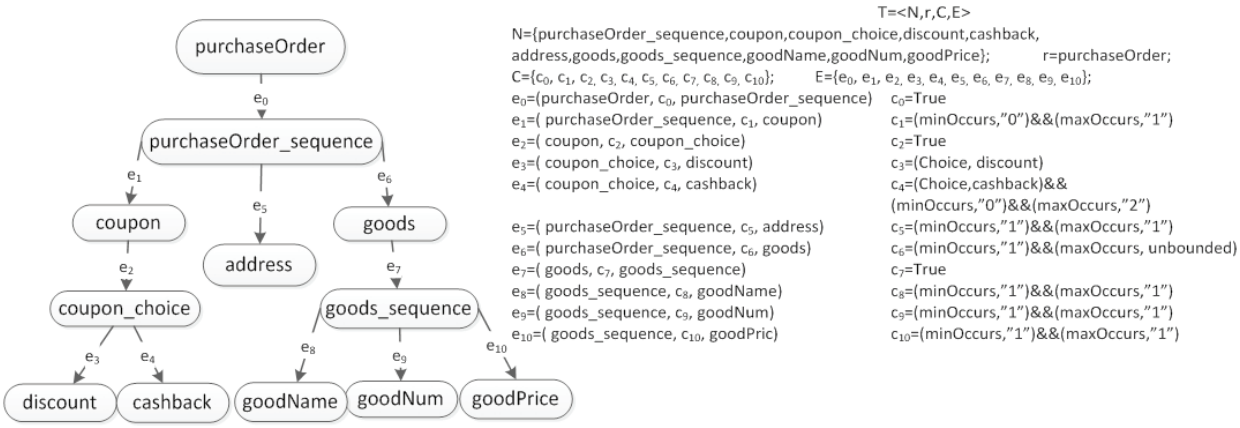

Fig. 3. PurchaseOrder Type Tree

4. Reducing Scenarios by Combinatorial Approach

4.1. Combinatorial Approach

Combinatorial approach [4] has been used to resolve the combinatorial testing issue caused by multiple factors, for instance, system configuration parameters, user inputs and other external events. The approach effectively checks the interaction among multiple factors by using fewer test cases.

Following are basic concepts involved in this approach.

- Variable: the input of software
- Level: the possible values of variable
- Strength: the degree of interaction among variables. For instance, strength is 2, which means the combination of any two variables.

Based on the variables, levels and required strength, combinatorial testing approach can generate a set of test cases by variable combinations.

4.2. $Cascade$

Cascade [5] is a combinatorial testing tool implements following functions:

- variable type: support string, integer and double types
- strength: user can assign a range of strengthes
- constraits: support comparison among variable, value and string; support logical and arithmetic operators, etc.

A constraint expression in $Cascade$ takes the form:

$$expA \longrightarrow expB$$

Which means expression $expA$ implies expression $expB$.

The input of $Cascade$ tool includes: (1) a set $\{(V_i, L_i) \mid i = 1, 2, 3, \ldots, n\}$, where V_i is a variable, and L_i is a set of levels of the variable V_i. Here n is natural numbers. (2) a set of constraint expressions $Cons$. The output of $Cascade$ is a set of value combinations expressed as $\{\langle l_1, l_2, \ldots\rangle | l_1 \in L_1, l_2 \in L_2, \ldots, l_n \in L_n\}$.

4.3. XML Schema Type Tree

In order to easily process XML Schema type definition, we model an XML Schema type structure and its indicator constraints as a tree. Instead of focusing on the constraining facets for built-in data types, we only focus on the indicators that can lead to the diversity of type.

A XML Schema type tree T is a 4-tuples:

$$T = \langle N, r, C, E\rangle$$

where:

- N is a finite set of element nodes and control nodes such as $sequence$ and $choice$.
- r is a root node.
- C is a finite set of indicator constraints between nodes in $N \cup \{r\}$ and nodes in N. In this paper, the indicator that we mainly focus on are $minOccurs$, $maxOccurs$ and $Choice$.
- E is a finite set of edges. An edge can be expressed as $e(m, c, n)$, where $c \in C, m \in N \cup \{r\}$ and $n \in N$.

A indicator constraint in C may take either one of following basic constraints or the conjunction of them.

- $True$
- $\langle minOccurs, n\rangle$ $(n = 0, 1, 2, \ldots, m)$, which requires the minimum number of times an element can occur is n. Here m is a nature number.
- $\langle maxOccurs, n\rangle$ $(n = 0, 1, 2, \ldots, unbounded)$, which requires the maximum number of times an element can occur is n, which can be assigned as either a nature number or $unbounded$.
- $\langle Choice, b\rangle$, which requires b is one of branches of $Choice$ indicator.

Figure 3 presents the tree model of $purchaseOrder$ type. The root node is $purchaseOrder$. There are three control nodes $puchaseorder\text{-}sequence$, $goods\text{-}sequence$ and $coupon\text{-}choice$.

4.4. Transform and Example

In order to use Cascade tool for effectively combining multiple values of multiple variables, we do the transformations from XML schema type tree to input model of Cascade, and from output model of Cascade to XML Schema type tree. Due to space constraints, we omit the algorithm details here.

As shown in Section 2, the successful purchasement scenario has ten interactions. The variable types are defined using XML Schema indicators. After type partition, the input model of $Cascade$ has 6 variables: $coupon$,$coupon\_choice$, $cashback$,$goods$,$bankInfo$ and $bankID$. Table 1 shows the variables and their levels. Totally, there are $2*2*3*2*3*3 = 216$ combinations without reduction.

We use $Cascade$ to reduce the combinations. In our experiment, we assume $strength = 2$ and constraints are:

$$bankInfo = ICBC \longrightarrow bankID = ICBC\_ID$$

$$bankInfo = CBC \longrightarrow bankID = CBC\_ID$$

$$bankInfo = BC \longrightarrow bankID = BC\_ID$$

After reducing, we get only 10 combinations. The result is shown in table 2. In practical application with more factors, the approach will be more effective.

TABLE 1. The Example Input Model of $Cascade$

variable	level1	level2	level3
coupon	0	1	
coupon_choice	discount	cashback	
cashback	0	1	2
goods	1	2	
bankInfo	ICBC	CBC	BC
bankID	ICBC_ID	CBC_ID	BC_ID

TABLE 2. The Example Output of $Cascade$

coupon	coupon_choice	cashback	goods	bankInfo	bankID
0	discount	0	2	ICBC	ICBC_ID
1	cashback	1	2	BC	BC_ID
0	cashback	2	1	CBC	CBC_ID
1	discount	0	1	BC	BC_ID
1	discount	2	2	CBC	CBC_ID
1	cashback	1	1	ICBC	ICBC_ID
0	discount	1	2	CBC	CBC_ID
0	cashback	2	2	BC	BC_ID
1	cashback	0	2	CBC	CBC_ID
1	cashback	2	2	ICBC	ICBC_ID

Each line of table 2 can be transformed to generate a new set of type trees with no uncertain type. And in the point of view of combinatorial testing, the result set of subtypes assurances the combination of any two indicator factors appears at least once. Test data generated based on it can be used to evaluate how well the web services under test validate and process the XML massages.

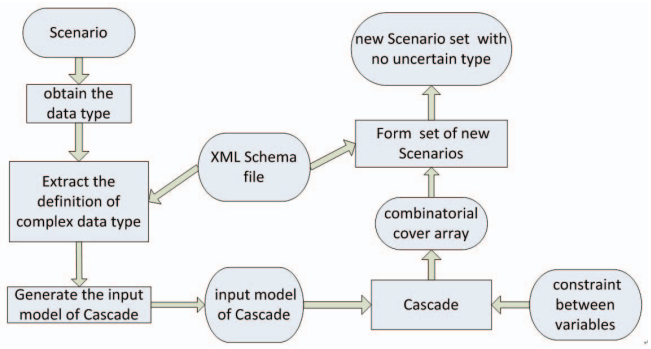

Fig. 4. The Process of Implemented Tool

5. Conclusion and Future Work

We implement a proof of concept toolkit to implement our approach ,the process is shown in figure 4.

A choreography specification identifies allowable ordering of message exchanges in a distributed system. How to guarantee the conformance between choreography specification and its implemented Web services is a challenge question in service-oriented computing field. It has attracted much of work. This paper presents our first step towards testing conformance based on choreography scenario.

Since a scenario may involve multiple variables, and each variable may take different subtypes (or values), it may cause the number of combinations of variable values is very big, which means the number of generated testing scenarios is also very big, and finally increases the cost of testing. The main contribution of this paper includes:

- It presents a scenario based testing approach for guaranteeing the conformance between choreography and its implemented Web services.
- It tests conformance of choreography from both data and control views.
- It first uses combinatorial testing approach to reduce choreography scenarios.

For the future work, on one hand, we intend to continue the development of our tool, on the other hand, to implement test date generation based on testing scenarios from this work.

References

[1] "Web Services Choreography Description Language (WS-CDL), version 1.0," 2005, http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/.

[2] W. Pacharoen, "Conformance verification between web service choreography and implementation using learning and model checking," in *IEEE International Conference on Web Services (ICWS)*, 2011.

[3] J. Eder, M. Lehmann, and A. Tahamtan, "Conformance test of federated choreographies," in *Proceedings of the 3th International Conference on Interoperability for Enterprise Software and Applications*. Springer, 2007, pp. 223–234.

[4] D. Cohen, I. C. Society, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The aetg system: An approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, pp. 437–444, 1997.

[5] "Cascade: CAS Covering Array DEsigner," http://lcs.ios.ac.cn/~zhangzq/ct_toolkit.html.