Test case generation for WS-BPEL from a static call graph

Wareerat Bousanoh and Taratip Suwannasart
Department of Computer Engineering, Faculty of Engineering
Chulalongkorn University, Bangkok, Thailand

Email: Wareerat.Bo@student.chula.ac.th, Taratip.S@chula.ac.th

Abstract. Web service is reusable, easy to maintain, widely used, and very popular. However, web service development is not adhered to any development languages in variety of platforms or frameworks. There must be a standard in controlling and coordinating a business process through a web service. This standard is called WS-BPEL. Web service testing is an important process in software development to meet the needs of business process. Recently some researches proposed concepts and tools used to generate test case for WS-BPEL, while they focus only a business processes within a single WS-BPEL file. This paper proposes an approach for generating a WS-BPEL test cases from a static call graph to support the case which has a calling subprocess between WS-BPEL files. The generated test cases are compiled with test paths based on the branch coverage.

1. Introduction
Nowadays, web service is very popular, due to its reusable and easy to maintain. A web service is designed to support interoperable between software applications over a network. Web service development is one of the software development under the principle of Service Oriented Architecture (SOA) [1]. SOA is about dividing and making software components independence. However, web service development is not adhered to any development languages. For that reason, there must be a web standard in controlling and coordinating a business process through a web service. The standard is called Web Services Business Process Execution Language (WS-BPEL), which is developed by OASIS organization [2].

WS-BPEL is a language for describing a behaviour of a business process based on interactions between the process and its partners. The interaction with each partner occurs through web service interfaces [2]. A WS-BPEL process represents all partners and interactions with these partners in terms of abstract WSDL interfaces [2]. WSDL is used to describe web services and tells a client how to compose a web service request and describes the interface that is provided by the web service provider.

Web service testing is an important process in software development to meet the needs of business process. It also reduces the number of software errors before delivery and ensures that a software is correct. Recently, there is a research which proposes concepts and a tool to generate test cases for WS-BPEL, while it focuses only a business processes within a single WS-BPEL file. In case of calling a subprocess between WS-BPEL files, a static call graph is used to help testers to understand WS-BPEL files and related variables in test paths for generating test cases. Otherwise, testers must be proficient in WS-BPEL files for generating test cases.
This paper proposes an approach for generating WS-BPEL test cases from a static call graph to support the case which has a calling a subprocess between WS-BPEL files. The generated test cases are complied with test paths based on the branch coverage.

2. Related work

There have been researches attempting to generate test cases for WS-BPEL. P. Nakngern and T. Suwannasart [3] proposed a design of WS-BPEL test case generation tool based on path conditions, which selects test paths from the control flow graph and map all related variables with random values from path conditions and their constraints. Ebrahim Shamsoddin-Motlagh [4] presented an approach for test cases generation automatically at the SOA system, which creates a control flow graph of BPEL file in the system and services related to the main service. Then, the test cases are created randomly for primary test from the graph in the generated system. S. Laokok and T. Suwannasart [5] proposed an approach to generate test cases from a static call graph in order to cover all class interfaces based on branch coverage.

3. Background

3.1. WS-BPEL

Web Service Business Process Execution Language (WS-BPEL) is the language for determining a model describing behavior of business processes which are coordinated between users and related web services [2]. It is a standard language developed by OASIS, and its distribution started in 2007 in a form of XML structure.

A business process of WS-BPEL is a reusable by using subprocess extension [6]. A subprocess is a part of BPEL code that is separated from a process which has the same behavior. The subprocess extension has the following benefits:

- to reduce a duplicate process.
- easy to modify and maintain.
- to reduce memory usage, especially for a complex process.

In this paper, we support for two types of subprocesses, which are inline subprocess and standalone subprocess.

1) An inline subprocess is defined as a part of BPEL 2.0 process at the <process> level [6].

2) A standalone subprocess is the process defined in a file with extension .sbpel (subprocess BPEL extension) [6].

3.2. Static call graph

A static call graph is a graph which represents the relation between methods [7]. Each method in a program is a node where a relationship between nodes is an edge. An activity in BPEL files is similar to a method in the general program, thus each node represents an activity in BPEL files and an edge represents calling of a BPEL file. For example, figure 1 shows a call graph for a simple Java program that has method A calling method B, method B calls method C, and method C calls methods B and D.

Figure 1. An example of a static call graph in Java program [7].
4. Methodology
This paper presents an approach for generating WS-BPEL test cases using a static call graph. The framework of our approach is shown in figure 2 and is described in subsections as follows:

4.1. Import files
Testers import files which consist of WS-BPEL, WSDL, and XSD files.

4.2. Generate test paths from a static call graph
This process has four steps. Firstly, all XML tags in WS-BPEL files are read to generate a static call graph in the next step. The WS-BPEL example file, CreditCardPay.bpel, a promotion of credit card payment in a shopping mall, which is the initial WS-BPEL file. The XML tag <bpx> in CreditCardPay.bpel file calling the sbp_VISARewards.bpel file is shown in figure 3.

```
<assign name="assign_VISADiscount"> 
  <copy>
    <from>
      <importVariable name="client.Price"/>
    </from>
    <to>
      <assign name="assign_VISADiscount" expression="(a*ImportVariable.payload/client.Price)/100"></assign>
    </to>
  </copy>
</assign>
<extensionActivity>
  <op:default xmlns:op="http://xmlns.oracle.com/904Cdemo/CreditRewards/sbp_VISARewards" target="op1:sp_VISARewards"/>
</extensionActivity>
```

Figure 3. The Initial WS-BPEL file. (CreditCardPay.bpel)

The XML tag <bplex> in sbp_VISARewards.bpel file which calls the sbp_VISAGift042018.sbpel file is shown in figure 4, and the sbp_VISAGift042018.sbpel file is shown in figure 5.

```
<assign name="assignCashBack1000"> 
  <copy>
    <from>1500</from>
    <to expression="urn:oaasis:names:tc:wspb:2.0:mscl:output:CashbackAmount.Output"></to>
  </copy>
</assign>
<extensionActivity>
  <bplex xmlns:sp="http://xmlns.oracle.com/904Cdemo/CreditRewards/sbp_VISAGift042018" target="sp1:sp_VISAGift042018"/>
</extensionActivity>
```

Figure 4. The sbp_VISARewards.bpel file.
Secondly, each WS-BPEL activity from CreditCardPay.bpel, sbp_VISARewards.sbpel, and sbp_VISAGift201804.sbpel file are analyzed to generate a control flow graph. For example, the control flow graph of CreditCardPay.bpel file, sbp_VISARewards.sbpel file and sbp_VISAGift201804.sbpel file are shown in figure 6, 7, and 8 respectively.

Figure 5. The sbp_VISAGift201804.sbpel file.

Figure 6. The control flow graph of CreditCardPay.bpel file.

Figure 7. The control flow graph of sbp_VISARewards.sbpel file.

Figure 8. The control flow graph of sbp_VISARewards.sbpel file.
Thirdly, a static call graph is generated from each control flow graph. The example in figure 9 presents control flow graphs of each activity and created static call graph of business process.

![Control Flow Graphs](image)

**Figure 9.** Control flow graphs of each activity and created static call graph of business process.

Finally, test paths are generated from the created static call graph and stored into the database. From the static call graph in figure 9, there have five test paths covering branch coverage as follows:

- **P1:** A1 – A2 – A3 – A4 – B1 – B2 – B3 – C1 – C2 – C3 – C4 – B8 – A7
- **P2:** A1 – A2 – A3 – A4 – B1 – B2 – B3 – B6 – C2 – C4 – B8 – A7
- **P3:** A1 – A2 – A3 – A4 – B1 – B2 – B5 – B6 – B8 – A7
- **P4:** A1 – A2 – A3 – A4 – B1 – B2 – B5 – B7 – B8 – A7
- **P5:** A1 – A2 – A3 – A6 – A7

4.3. **Analyze the variables and their constrains**

The variables and their constrains of each node of the static call graph are analyzed using the WSDL and XSD files which are imported from process A and the test paths generated from process B. Then, it is stored into the database.

4.4. **Instrument the WS-BPEL files**

In this process, the WS-BPEL files are instrumented to verify if all test cases cover all branches in the static call graph. At the end of this process, a tester will receive the instrumented WS-BPEL files to be executed through the BPEL engine.

4.5. **Generate test cases**

The test paths, variables and constrains which are outputs of process B and C and used to generate test cases based on branch coverage. The test input values are randomized by using the variable constrains. At the end of this process, a tester will receive the test cases to be executed through the BPEL engine.

4.6. **Generate test reports**

After the tester executes the instrumented WS-BPEL files and the test cases through the BPEL Engine, test logs are recorded by BPEL Engine. Finally, the test reports are generated using the test data from the previous processes.

5. **Conclusion**

This paper introduces a design of test case generation for WS-BPEL from a static call graph based on branch coverage. We propose an approach for generating test case that supports the case of calling subprocess between WS-BPEL files. In the future, we will aim to apply this approach in developing a tool from our proposed approach.
References

[1] “Reference Architecture Foundation for Service Oriented Architecture Version 1.0”. [Online]. Available: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html [Accessed: 24 January 2018]

[2] “Web Services Business Process Execution Language Version 2.0”. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html [Accessed: 14 February 2018]

[3] P. Nakngern and T. Suwannasart (2017). A Design of WS-BPEL Test Case Generation Tool Based on Path Conditions. International MultiConference of Engineers and Computer Scientists.

[4] E. Shamsoddin-Motlagh (2013, October). Automatic Test Case Generation for Orchestration Languages at Service Oriented Architecture. International Journal of Computer Applications, Vol. 80, No7, pp. 35-44.

[5] S. Laokok and T. Suwannasart (2017). An Approach for Test Case Generation from a Static Call Graph for Object-Oriented Programming. International MultiConference of Engineers and Computer Scientists.

[6] “Oracle® Fusion Middleware Developer's Guide for Oracle SOA Suite”. [Online]. Available: https://docs.oracle.com/middleware/1221/soasuite/develop/GUID-FAA53226-470D-4120-95ED-1EAB6D96AA3F.htm [Accessed: 14 February 2018]

[7] “Call Graph Construction Algorithms Explained”. [Online]. Available: https://benholland.com/call-graph-construction-algorithms-explained/ [Accessed: 14 April 2018]