A Design of WS-BPEL Test Case Generation Tool Based on Path Conditions

Preecha Nakngern and Taratip Suwannasart

***Abstract*—In recent years, Service Oriented Architecture has gained significant attention, especially for web services. Also, WS-BPEL programs have become popular, as they can be used to create complex business processes by combining multiple web services together. This is one kind of WS-BPEL feature brings challenge into testing. There are many approaches and tools for automatic test case generation, such as graph-search based [5], random [6] or Message-Sequence [7] technique. But previous researches have not studied test case generation based on test path conditions. This paper, we propose a design of WS-BPEL test case generation tool based on path conditions, which can generate test cases and verify their coverage.**

***Index Terms*— WS-BPEL, WSDL, Test Case, Test Case Generation Tool**

I. INTRODUCTION

NOWADAYS, Service Oriented Architecture (SOA) [1] and web services have gained significant attention due to their capability to separate software into loosely-coupled parts called services, each of which does only one purpose. With this characteristic, SOA has the benefit of reusability and maintainability, especially for web services that are independent of language or environments.

The Web Services Business Process Execution Language (WS-BPEL) [2] has become a standard for web services orchestration, due to the ability to specify and execute business processes among web services. WS-BPEL is used to describe the logic for executing multiple web services in the business flow. Typically, WS-BPEL is used to describe a complex business process invoking dozens of web services are difficult to understand and test.

Currently, many tools exist for testing WS-BPEL to reduce testing time by automating test case generation [5], [6], [7], [8], and [9]. In those research works, we found use of random and manually specific value for a test case, so that it may take more time to create random values suitable for a target test path and may be error-prone. However, we can use benefits from WS-BPEL that have WSDL [3] documents describing details of WS-BPEL operations and its input constraints. We can use the WSDL documents to generate values of test case variables and use test path conditions from WS-BPEL to specify variables values for each test path. In this paper, we present a design of WS-BPEL test case generation tool based on path conditions that provides branch coverage of WS-BPEL.

P. Nakngern and T. Suwannasart are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand. e-mail: Preecha.Na@student.chula.ac.th, Taratip.S@chula.ac.th

The rest of the paper is organized as follows: Section II presents a background on relevant terms and testing concepts used in this paper. Section III analyses current approaches and tools for generating test cases for WS-BPEL applications. Section IV presents the structure and behavior of our tool followed by conclusions and future work in Sections V.

II. BACKGROUND

A. WS-BPEL

WS-BPEL is a language based on XML proposed by OASIS, used for a standard of web services business process execution. WS-BPEL is designed for describing business process in both abstraction and executable processes. For specifying business process behavior, WS-BPEL provides many activities, which are classified into 2 classes. First, basic activities such as *invoke*, *receive*, *reply*, *assign*, *wait*, etc., describe elemental steps of process behavior. Second, structured activities such as sequence, *if*, *while*, *flow*, *pick*, etc., prescribe the order of execution of basic activities to describe how a business process works. Through the specification of web service execution, WS-BPEL defines the relationship between processes with *partnerLink* and used WSDL to specify how they communicate.

B. WSDL

WSDL (Web Services Description Language) is a language based on XML, used for describing web service information. WSDL describes web service information in 2 types, 1) abstract type: operation and message information, described by PortType, Operation, Message, and Types, where PortType is a collection of operations, Operation is an action of web service, Message describes communication data structure, and Types is a data type definitions. 2) concrete type: protocol and data format specifications, described by Service, Port, and Binding, where Service is a collection of web service endpoints, Port describe endpoint by combine binding and network address, Binding are protocol and data specification. In this paper, we focus on abstract type information for use in analyzing variable constraints.

C. Software Testing

Software testing [4] is the process of evaluating and verifying the software to ensure that the software meets software specification and requirements. Software testing has 2 main approaches. First, functional testing (Black-Box Testing) is a test process based on the software specification, which a software function is defined in terms of inputs and

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

outputs. Second, structural testing (White-Box Testing) is a test process based on the software implementation. Structural testing focuses on test coverage of test cases, for which more test coverage means more chance of finding faults in the software. Test coverage of software can be divided into 3 levels.

1. Statement Coverage is coverage for every line of code that needs to be executed.
2. Branch Coverage is coverage for both the true and false conditions that need to be executed.
3. Path Coverage is coverage for every feasible path from a start node to a terminal node that needs to be executed.

III. RELATED WORK

In recent years, testing WS-BPEL has become a research focus in software testing. Yan et al. [5] use a graph-search based method to generate an extension of the control flow graph called BPEL flow graph (BFG) to represent a WS-BPEL program. BFG is used to create test cases by traversing the BFG. Theerapong and Twittie [6] propose a test suite for WS-BPEL, which can create test cases and create web service stubs for providing test cases that cover basis paths. The test case and web services are created by randomizing with constraints from a WSDL document and manual assignment. Moreover, instead of creating test cases from BPEL behaviors, Yitao et al. [7] propose a message sequence testing technique from OO programming to create a message-sequence graph (MSG), which can describe the order relation among messages in WS-BPEL. Chien-Hung et al. [8] use a combination of BPMN and WS-BPEL structures for testing. Samer et al. [9] generate test cases for testing web services based on a WSDL document.

Our approach covers generating test cases for WS-BPEL based on path conditions of WS-BPEL, by using input variable constraints from related WSDL documents, the approach does not require testers to create inputs manually. Test cases created by this tool can reach branch coverage if the condition's variables are tested by WS-BPEL.

IV. METHODOLOGY

In this section, we present concept, design, and functions of our test case generation tool. This tool contains 2 parts and includes 8 modules as illustrated in Figure 1. First, a test case generation part contains module A, B, C, D, and E. This part generates test cases by analyzing WS-BPEL, WSDL, and XSD files. Second, a test case verification part contains module F and G. This part verifies the coverage of test cases with coverage report from code instrumentation. To illustrate the tool behavior, an "employee information" process in WS-BPEL specification is used, "employee information" is a process to save employee information by receiving the client request, checking initial payload and response to the client. Figure 2 shows the design of employee information process.

A. Input receiver module

This module receives initial files from a tester. Initial files include WS-BPEL, WSDL, and XSD.

B. WS-BPEL input inspection module

This module analyzes the WS-BPEL file to find the initial node and its input variables, before analyzing relevant variables and their constraints, such as type, length, or min-max value from the WSDL and XSD files. The WSDL and XSD files are selected from the WS-BPEL parent link. Then a list of variables and its constraints is created to prepare for mapping to the test path.

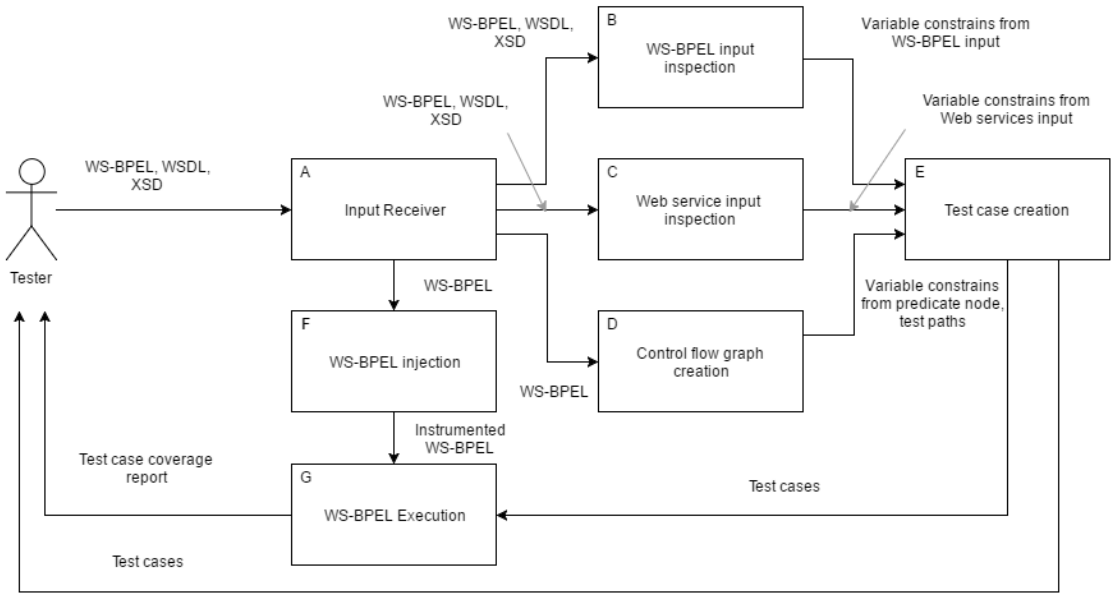

Figure. 1. Structure of the tool.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

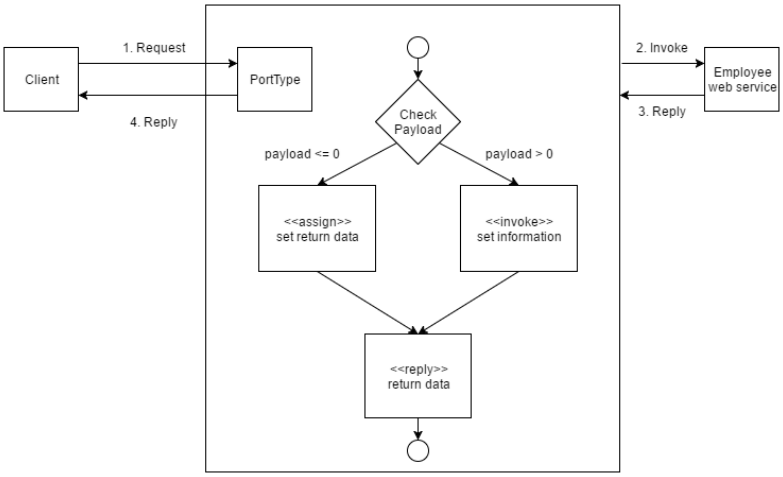

Figure. 2. Design of employee information process.

C. *Web service input inspection module*

This module analyzes the WS-BPEL file for all specified web services. Each web service must have a partner link to its WSDL and XSD files. Both files are then analyzed for web service input variables and their constraints. These variables will be merged with the WS-BPEL input variables from the previous module.

D. *Control flow graph creation module*

This module creates a control flow graph from analyzing WS-BPEL file. WS-BPEL has 2 types of activities. First, basic activities such as receive, invoke and assign are represented by a node in the control flow graph. Second, structured activities such as sequence, if, and while are represented by a predicate node or edge depending on its types. Table 1 lists the WS-BPEL activities and their control flow graph representation. Figure 3 shows the control flow graph of employee information process. After creating a control flow graph, we create test paths using breadth-first search. Figure 4 shows test paths of employee information process.

TABLE 1
A LIST OF WS-BPEL ACTIVITY MAPPED TO CONTROL FLOW GRAPH ELEMENT.

WS-BPEL activity	control flow graph element
basic type activity	
invoke	node
receive	node
reply	node
assign	node
throw	node
wait	node
empty	node
rethrow	node
exit	node (terminal node)
structured type activity	
scope	edge
sequence	edge
flow	edge
if	predicate node
while	predicate node
repeatUntil	predicate node
pick	predicate node
forEach	predicate node

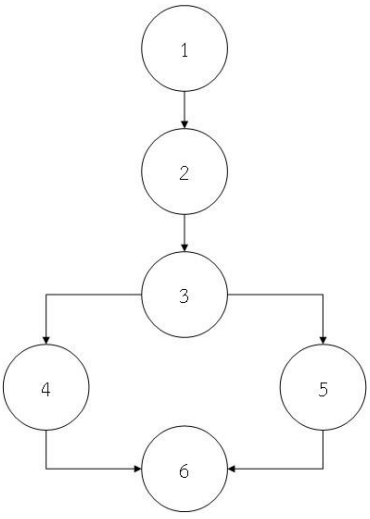

Figure. 3. The control flow graph of employee information process.

Path A. 1 – 2 – 3 – 4 – 6

Path B. 1 – 2 – 3 – 5 – 6

Figure. 4. Test paths of employee information process.

E. *Test case creation module*

To create a test case, we select one of test paths from the control flow graph and map all related variables with random values from path conditions and their constraints, then continue looping until completing all feasible test paths. Test cases created from this module are in XML format. Examples of test cases are shown in figure 5 - 6.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
<EmployeeRequest>
    <payload> 1 </payload>
    <employee>
        <FirstName> String </ FirstName>
        <LastName > String </ LastName>
        <Departement > String </ Departement>
        <Age> 50 </ Age>
    </employee>
</EmployeeRequest >
```

Figure. 5. Example of test case path A.

F. WS-BPEL injection module

To verify coverage of test cases, we provide a report about nodes reached by each test case. To make the report, we use code instrumentation to inject a logging function into WS-BPEL. The logging function tracks which nodes are reached by each test case.

G. WS-BPEL execution module

This module uses an instrumented WS-BPEL to execute test cases from the previous module, to show the coverage of a test case with logging messages as a report. Figure 7 shows an example of report generated by this module.

Test case No.	1
Execution Log	
======= Start Execution ======= Node No. 1 => Activity: receive. Node No. 2 => Activity: assign. Node No. 3 => Activity: if. Node No. 5 => Activity: invoke. Node No. 6 => Activity: reply. ======= End Execution =======	

Figure. 7. Example of test cases coverage report.

V. CONCLUSION

We propose a design of test case generation tool for WS-BPEL based on path conditions. This tool can generate test cases with branch coverage and verify their coverage, by analyzing the WS-BPEL structure and use it to create a control flow graph, then analyze WSDL and XSD for specific variable constraints used in WS-BPEL. Finally, variables are mapped to a path in the control flow graph for each test path to create a test case. Moreover, the tool can generate test case only if variables used in the path conditions are WS-BPEL inputs. The benefit of this tool is acquiring test cases for WS-BPEL with branch coverage without using higher testing skills, as well as time and cost saving.

For future work, we will apply this design to develop a prototype of the tool and test it with industry business processes.

```
<EmployeeRequest>
    <payload> 0 </payload>
</EmployeeRequest>
```

Figure 6. Example of test case path B.

REFERENCES

[1] "Reference Architecture Foundation for Service Oriented Architecture Version 1.0", 2012. [Online]. Available: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html. Last Accessed: 4 JAN 2017.

[2] "Web Services Business Process Execution Language Version 2.0", 2007. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf. Last Accessed: 4 JAN 2017.

[3] "Web Services Description Language (WSDL) 1.1", 2001. [Online]. Available: https://www.w3.org/TR/wsdl. Last Accessed: 4 JAN 2017.

[4] Paul C. Jorgensen, Software Testing: A Craftsman's Approach. Second Edition. CRC Press, 2002.

[5] Y. Yuan, Z. Li and W. Sun, "A Graph-Search Based Approach to BPEL4WS Test Generation", Software Engineering Advances, International Conference on, Tahiti, 2006, pp. 14-14.

[6] Theerapong Lertphumpanya and Twittie Senivongse, "Basis path test suite and testing process for WS-BPEL", WSEAS Transactions on Computers, Vol. 7 Issue 5, May 2008, pp. 483-496.

[7] Y. Ni et al., "Effective Message-Sequence Generation for Testing BPEL Programs," in IEEE Transactions on Services Computing, vol. 6, no. 1, First Quarter 2013, pp. 7-19.

[8] C. H. Liu, S. L. Chen, and X. Y. Li, "A WS-BPEL Based Structural Testing Approach for Web Service Compositions," 2008 IEEE International Symposium on Service-Oriented System Engineering, Jhongli, 2008, pp. 135-141.

[9] S. Hanna and M. Munro, "An Approach for Specification-based Test Case Generation for Web Services," 2007 IEEE/ACS International Conference on Computer Systems and Applications, Amman, 2007, pp. 16-23.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)