2009 33rd Annual IEEE International Computer Software and Applications Conference

An Improved Test Case Generation Method for Web Service Testing from WSDL-S and OCL with Pair-wise Testing Technique

Siripol Noikajana and Taratip Suwannasart
Software Engineering Laboratory
Center of Excellence in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
Siripol.N@student.chula.ac.th, Taratip.S@chula.ac.th

Abstract

Web Services are popular solutions that have been in recent years to implement software in every business domain. They provide an effective way to reuse functionality, which reduces development time and cost, and increases software reliability. Software testing is a key issue of Web Services development in order to ensure quality of services. Software testing aims to discover faults or defects in software to determine where its behaviors do not correspond with software requirements or test case specifications. Software testing is used to evaluate functional correctness, reliability, robustness, and performance. We describe a Web Service contract with a Web Service Semantics Language (WSDL-S) and the Object Constraint Language (OCL). WSDL-S can be used to identify pre- and post- conditions of Web Service operation by referring service rule from OCL. This paper presents an approach for generating Web Service test cases using WSDL-S and OCL, while test case generation method is a pair-wise testing technique.

1. Introduction

As the internet increases in importance in the global society and economy, the need to provide effective functionality across the Web has become more critical than ever before. One widely used technique to provide functionality via the Internet is through Web Services. However, the development of Web Services is a particularly difficult task due to the complexity of the environment in which they must function. One of the most difficult aspects of Web Service development is the complexity involved in conducting effective system testing. In fact one of the most difficult challenges in the discipline of software testing is how to effectively test Web Service software [1-10]. Therefore, much research has been done to improve software testing of Web Services.

Our main categories of generating Web Service test cases are generating Web Service test cases by using WSDL [2, 4, 7, 10] and generating Web Service test cases by adding information from WSDL [1, 3, 5, 6, 8, 9]. Bertolino *et al* [7, 10] proposed generating Web Service test cases by analyzing XML schema documents that are specified in WSDL. Siblini *et al* [4] proposed applying the mutation testing technique to WSDL standard for testing Web Services. He identified nine mutation operators; STCE, STCA, OTCE, OTCA, STEN, STSE, STSA, SMP, and SPM. The first six mutation operators are used to mutate the Types element of WSDL. SMO mutation operators are used to mutate the Messages element. Finally, SPM mutation operators are used to mutate the PortType element. Tsai et al [6, 9] proposed generating Web Service test cases by adding information from WSDL standard: input-output dependency, invocation sequences, hierarchical functional description, and sequence specifications. They defined a method where a test engine can retrieve information which is necessary for generating Web Service test cases.

A Web Service can be identified by a URI (Uniform Resource Identifier). The interfaces and bindings of a Web Service can be discovered, defined, and described, as XML artifacts according to a Web Services Description Language (WSDL) [11]. WSDL can be used to describe Web Service operations including inputs, outputs, and exceptions.

0730-3157/09 $25.00 © 2009 IEEE
DOI 10.1109/COMPSAC.2009.25

Unfortunately, WSDL cannot identify pre- and post-conditions of Web Services. Therefore, the W3C has proposed a Web Service Semantics Language (WSDL-S) that is an extension of WSDL. WSDL-S addresses the problem of the syntactic level and the lack of semantic expressiveness that needed to represent the requirements and capabilities of Web Services. WSDL-S adds semantic annotations to WSDL in order to describe the requirements and capabilities of Web Services.

Pre- and post- conditions of Web Services can be identified using WSDL-S, however a detailed description of the pre- and post- conditions must utilize additional descriptive languages. Explicit descriptions of pre- and post- conditions of a Web Service operation can be accomplished using either the Semantic Web Rule Language (SWRL) or the Object Constraint Language (OCL). SWRL is not as powerful as OCL. It can be only used to express a conjunction of conditions using AND logical expression. OCL is more expressive and can describe more general expressions of pre- and post-conditions.

One of the most daunting tasks of software testing is the difficulty of conducting exhaustive validation and verification tests. The number of combinations quickly becomes astronomical. Consider, for example a system with ten different input parameters with each input parameter allowed to have five possible values. Exhaustive combinatorial testing will require 5^{10} tests be conducted. No projects have enough time or budget to permit such exhaustive testing.

The use of combinatorial covering design [12] is one popular method for performing software testing. Combinatorial covering test design is based on the theory that most software faults are caused by a relatively few combinations of input parameters. Combination strategies are test-case selection methods where test cases are identified by combining values of the different input parameters based on some combinatorial strategies. An N-way testing technique is defined as a set of tests for N parameters, so that every combination of their valid values are covered by at least one test case. The failure-triggering fault interaction (FTFI) number [13] was defined as the number of input parameter combinations required to trigger a failure. For example if a certain failure is caused by the interaction of two input parameters, then the FTFI number would be two. Figure 1 presents the empirical results of the research for four different types of software systems. It plots cumulative error detection as a percent of the FTFI number. As can be seen, 29% to 68% of total faults are caused by single parameters, i.e. N=1 [13].

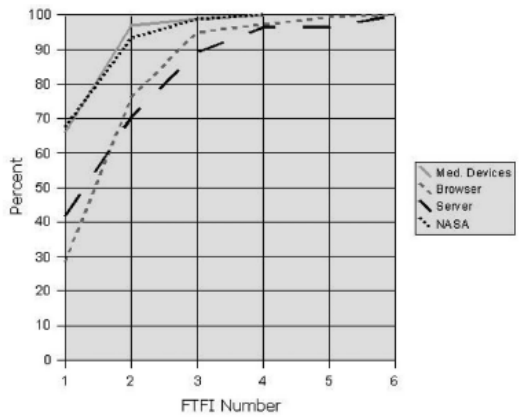

Figure 1 The cumulative distribution percent of faults triggered by n-way Conditions [13].

The implications of this research are important in the development of a test case selection technique. Since up to 95% of defects can possibly be uncovered by pair-wise testing technique, test case generation based on pair-wise strategy may be adequate for many types of software systems.

In previous work [3, 8], we have proposed techniques for generating Web Service test cases. The work focused on Web Service contracts based on specifications using WSDL-S, and SWRL. However, the earlier work was based on a single-parameter fault assumption, and only identified selected pre-and post-conditions. In this paper, we utilize both OCL and WSDL-S to improve identifying service rules. In addition we apply a pair-wise testing technique to generate Web Service test cases including identifying specific expressions of pre- and post- conditions that include OR logical expression. OCL identifies service rules that can be used to express pre- and post-conditions of Web Service operation. We can express pre- and post- conditions of Web Service operation from WSDL-S by describing service rules in OCL.

The remainder of this paper is organized as follows: section 2 introduces the background related to the Object Constraint Language, the Semantic Web Services, and pair-wise testing. Section 3 presents the approach for generating Web Service test cases. In Section 4, we present an example. Finally, conclusions are drawn and future work is described in Section 5.

2. Background

This section introduces background material required for this paper. It provides an overview of the Object Constraint Language, the Semantic Web Services, and pair-wise testing.

2.1. Object Constraint Language

The Object Management Group (OMG) [14, 15] has presented the OCL specification. OCL is a formal language used to describe expression on UML models. These expressions typically specify invariant conditions that must hold for the system being modeled. In this paper, we use OCL expression to specify pre- and post- conditions of Web Service operations.

Figure 2 presents to identify the Rectangle types' specification for Web Services using OCL expression. In figure 2, *rectangleTypeRule* is one service rule which returns string data type. Its results have four string alternatives: square, diamond, rectangle, parallelogram, and convex. However, this service rule can be invoked when all conditions of pre-condition have been hold. *rectangleTypeRule* has 6 parameters: x, y, w, z, u and v. Four first parameters are side of rectangle and other parameters are diagonal.

```
context rectangleTypeRule (x:int, y:int, w:int, z:int,
u:int, v:int):string
 pre:  u > 0 and v > 0
       and w > 0 and x > 0 and y > 0 and z > 0 and
       w < 100 and x < 100 and y < 100 and z < 100
 post: if ((x == y) and (y == z) and
          (w == z) and (u == v))
       then
            result = "square"
       else if((x == y) and (y == z) and
            (w == z) and (u != v))
       then
            result = "diamond"
       else if((x == y) and (y != z) and
            (w == z) and (u == v))
       then
            result = "rectangle"
       else if((x == y) and (y != z) and
            (w == z) and (u != v))
       then
            result = "parallelogram"
       else if ((x != y) and (y != z) and (w != z))
       then
            result = "convex"
```

Figure 2. Specification of rectangle types in OCL document.

2.2. Semantic Web Services

The Semantic Web Service [16-20] is one paradigm to implement Web Services. This paradigm models implement the behavior of the service, and represent a contract governing the meaning and purpose of the interaction. W3C have presented a Web Service Semantics language (WSDL-S) to define a mechanism to associate semantic annotations with Web Services. WSDL-S represents the capabilities of Web Services by associating a semantic description to the Web Service in order to enable automatic search, discovery, selection, composition, and integration across heterogeneous users and domains.

WSDL-S is an extension language of WSDL by adding semantic annotations including three attributes, and two elements:

1. The precondition element is a set of assertions that must be met before Web Service can be invoked.
2. The effect element is an element that is a result of invoking a Web Service operation.
3. The modelReference attribute is a specification of association between WSDL entity and a concept in some semantic model. It can be combined with a complex type, element, or operation, as well as with the extension precondition and effect elements.
4. The schemaMapping attribute is a handling structure which differentiates between schema elements of Web Services and their corresponding semantic model concepts.
5. The category attribute is a service categorization of information for publishing a service in a Web Services registry.

Figure 3 represents RectangleType Web Service contract. RectangleType consists of one post-condition. This condition is expressed with service rules, defined in OCL specification as shown in figure 2.

2.3. Pair-Wise Testing Technique

A pair-wise testing technique [21, 22] is one method of combination strategies, which are commonly used as test case selection methods. The technique creates test cases by combining values of input parameters of the test object. Therefore, a pair-wise testing technique is a testing criterion based on specification, which requires each pair of parameters. Every combination of their valid values should be covered by at least one test case in the test suites.

3. Approach

The focus of this research project is Web Service test case generation by using semantic annotations. In our approach, we describe a Web Service contract using WSDL-S and express pre- and post- conditions

of services with OCL. We propose an approach to generate Web Service test cases by using pair-wise testing. Pair-wise testing can be used to detect 70 – 95 percent of defect reports. Pair-wise testing also requires substantially fewer test cases than other combination testing [13].

```
<types>
  <xsd:schemaelementFormDefault="qualified"
    targetNamespace="http://tempuri.org/">
    <xsd:element name="Rectangle">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="x"  type="xsd:int"/>
          <xsd:element name="y"  type="xsd:int"/>
          <xsd:element name="w" type="xsd:int"/>
          <xsd:element name="z"  type="xsd:int"/>
          <xsd:element name="u" type="xsd:float"/>
          <xsd:element name="v"  type="xsd:float"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="RectangleResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="RectangleResult"
              type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</types>
<message name="RectangleSoapIn">
  <part name="parameters"
      element="tns:Rectangle" />
</message>
<message name="RectangleSoapOut">
  <part name="parameters"
      element="tns:RectangleResponse" />
</message>
<portType>
  <operation name="RectangleType">
    <input message="tns:RectangleSoapIn" />
    <output message="tns:RectangleSoapOut" />
    <wssem:effect name="ResultEffect"
        wssem:modelReference=
        "rectangleTypeRule"/>          } post-condition
  </operation>
</portType>
```

Figure 3. RectangleType Web Service contract.

Figure 4 represents the activities for generating Web Service test cases. These activities include WSDL-S Analysis, Service Rule Analysis, Web Service Operation Specification Generation, and Test Case Generation. We will describe details of each activity in next section. The WSDL-S Analysis activity is first conducted which coincides with the Service Rule Analysis activity. The result of the WSDL-S Analysis activity and the Service Rule Analysis activity are preliminary Web Service operation definitions and service rule definitions respectively. These results are inputs to the Web Service Operation Specification Generation activity. The Web Service Operation Specification Generation activity binds conditions of operation definitions with service rule. After the Test Case Generation activity analysis input domain partitions of Web Service operation, this activity generate Web Service test cases by using pair-wise testing technique. The result of the Test Case Generation activity corresponds to Web Service Test Case specification (WSTC), as shown in Figure 5.

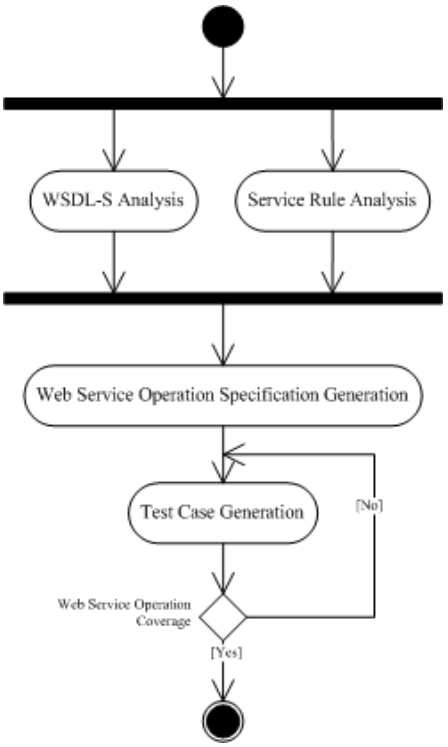

Figure 4. Web Service test case Generation Activity Diagram

3.1. WSDL-S Analysis

A WSDL-S document is used to describe a Web Service including semantic annotations which define pre- and post- conditions, service categorization, and so forth by adding new elements and attributes to WSDL, which defines these semantic annotations. It is

relatively easy to extend the existing capabilities of WSDL. Hence, WSDL-S does not define a language that spans across the different levels of the Web Service stack, rather it is limited to WSDL simplifying the task of providing semantic representation of services [3].

```
<element name="items">
  <complexType>
    <sequence>
      <element name="item" type="item"
          minOccurs="0"
          maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
<complexType name="item">
  <sequence>
    <element name="inputs" minOccurs="0"
          maxOccurs="unbounded" />
      <complexType>
        <sequence>
          <element name="input" type="variable_desc" />
        </sequence>
      </complexType>
    <element name="results" minOccurs="0"
          maxOccurs="unbounded" />
      <complexType>
        <sequence>
          <element name="result" type="string" />
        </sequence>
      </complexType>
  </sequence>
  <attribute name="operation" type="string" />
</complexType>
<complexType name="variable_desc">
  <sequence>
    <element name="variable" type="string" />
      <complexType>
        <complexContent>
          <restriction>
            <attribute name="dataType"  type="string" />
          </restriction>
        </complexContent>
      </complexType>
    <element name="value" type="string" />
  </sequence>
</complexType>
```

Figure 5. The Web Service Test Case specification (WSTC).

The WSDL-S Analysis activity uses semantic annotations in order to generate Web Service test cases including pre- and post- conditions as in our previous works [3, 8]. This activity has two analysis prescriptions including - a data type definition and an operation definition. Figure 6 illustrates the sub-activities of Web Service analysis.

A data type definition, which is defined with an XML schema, describes parameters of the inputs and outputs of Web Service operation. We can define data type definition with using either a simple type or a complex type of XSD schema or user built-in data types [2, 23]. A basic data type definition consists of the variable's name, data type, bound value (optional), and number of data value (optional). A data type either as variable, definition can be defined by three alternative formats - class or enumeration. Hence, a class is generated from either all elements or sequence elements. An enumeration is generated from either enumeration elements or choice elements.

An operation definition consists of the pre- and post- conditions, and the input and output parameters which are described by using WSDL-S. The pre- and post- conditions refer to the rule name. We describe how the pre- and post- conditions refer to rule name later. However, each Web Service operation has at most only one pre-condition. The input and output parameters are described to be the abstract message which is mapped using data type definitions.

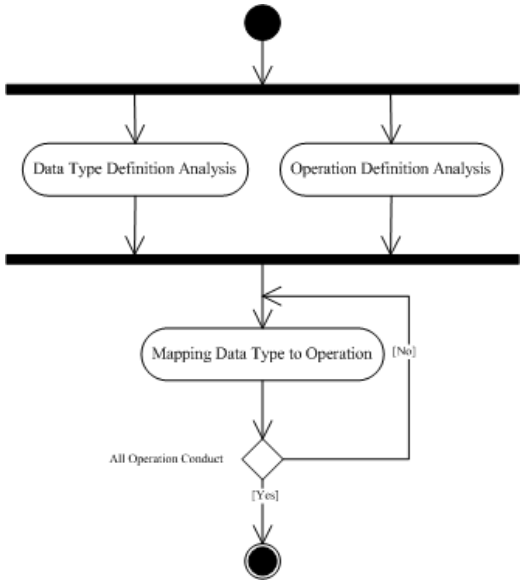

Figure 6. Sub-Activities of WSDL-S Analysis.

3.2. Service Rule Analysis

Input to the Service Rule Analysis activity is an OCL document. Service Rules are used to express pre- and post- conditions of a Web Service operation by using OCL as described in Figure 7. Figure 8 presents an algorithm to analyze OCL documents. The analysis results in service rules which are referenced by *modelReference* attribute of WSDL-S.

1. Define a service rule name using *context*.
2. Define invariant constraint of service rule using *inv*.
3. Each service rule can define pre- and post-conditions using *pre* and *post* respectively.
4. Refer the service rule name to the pre- and post-conditions defined in WSDL-S by using the *modelReference* attribute.
5. Variables must be defined according to Data type definitions.

Figure 7. The service rule syntax

1. Retrieve OCL documents specified by the WSDL-S document.
2. Analyze the OCL document
3. Traverse the service rule, where each item in the rule consists of invariant constraints, pre-, and post- conditions.
4. Repeat step 2 – 3 with all pertinent OCL documents.
5. Save the service rules to database.

Figure 8. The Service Rule Analysis algorithm

3.3. Web Service Operation Specification Generation

The inputs to the Web Service Operation Specification Generation activity are the set of service rules and the set of operation definitions. Figure 9 shows an algorithm to generating Web Service operation specification. The results are Web Service operation specifications that are binding data between service rules and operation definitions. Web Service operation specifications are independent.

1. Retrieve OCL documents specified by the WSDL-S document.
2. Analyze the OCL document
3. Traverse the service rule, where each item in the rule consists of invariant constraints, pre-, and post-conditions.
4. Repeat step 2 – 3 with all pertinent OCL documents.
5. Save the service rules to database.

Figure 9. The Service Rule Analysis algorithm

3.4. Test Case Generation

The Test Case Generation activity is consequence the Web Service Operation Specification Generation activity. The Test Case Generation activity is inspired with a pair-wise testing technique. This activity consists of three sub-activities: generating input parameter model (IPM), generating test cases with pair wise testing technique, and conflict test case elimination. An IPM consists of a set of modeling parameters with a set of values for each parameter. The IPM parameters are based on functions of the actual parameters of the system under test (SUT). An IPM parameter value represents a non-empty subset of values of the input space and possibly the state of the system under test [21]. Figure 10 presents a mechanism of the generating IPM that is analyzing Web Service operation specification. In this paper, the input parameter model of each Web Service operation are consequent upon boundary value, invariant constraints, statement ordering, pre-, and post-conditions. Next, this approach generates Web Service test cases using a pair-wise testing technique. However, test suite may be included conflict test cases. A conflict test case is an impossible combination of two or more IPM. Hence, conflict test cases are necessary automatically removed by using the replace method [21].

1. Retrieve Web Service operation specifications.
2. Analyze invariant constraints.
3. Traverse the invariant constraint, and each condition is entailed to generating IPM.
4. Analyze the pre-conditions.
5. Traverse the pre-condition, and at each statement:
    a. If statement is condition expression, then each condition is entailed to generating IPM.
    b. If statement is value assignment which some parameters has relation with consequent condition expression statements, then this statement is entailed to generating IPM.
6. Analyze the post-conditions.
7. Traverse the post-condition, and at each statement:
    a. If statement is condition expression, then each condition is entailed to generating IPM.
    b. If statement is value assignment which some parameters has relation with consequent condition expression statements, then this statement is entailed to generating IPM.
8. Repeat step 2 through 7 with all pertinent Web Service operation specifications.

Figure 10. The generating input parameter domain algorithm

4. Examples

This paper presents the results of two Web Service examples, RectangleType, and IncreaseDate. We evaluate the efficiency of Web Service test cases via mutation testing. We use relational operator replacement (ROR) as mutation operator. We illustrate step by step of generating Web Service test cases as follows:

The first Web Service is RectangleType service. We start to retrieve a rectangle types' specification, and a RectangleType Web Service contract as described in figure 2 and figure 3 respectively. Next, we analyze a rectangle types' specification, and Web Service contract. Next, we generate IPM, then afterwards to generate Web Service test cases using a pair-wise testing technique. The test suite does not contain any conflict test cases. Hence, the replace method is not invoked. As in figure 11, our approach contains 3 test cases and the test suit [24] contains 5 test cases.

As in figure 12 presents the percentage of mutants killed between our approach and decision-base testing [24], our approach kills 12 out of 19 mutants, and decision-base testing [24] kills 16 out of 19 mutants.

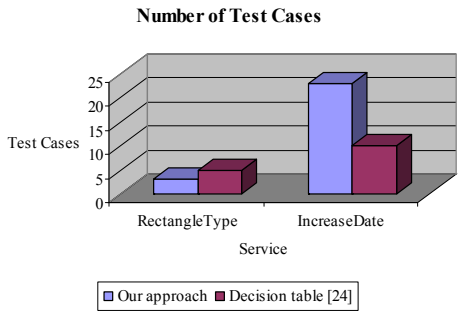

Figure 11. A number of Web Service test cases.

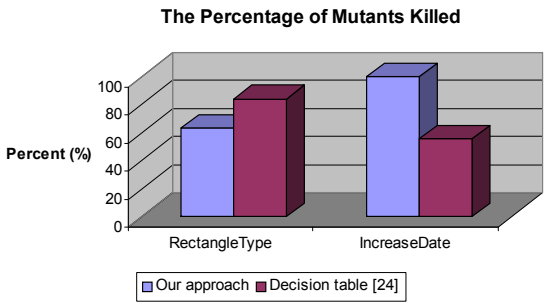

Figure 12. The percentage of mutants killed.

Next IncreaseDate service is invoked experience our approach, we first retrieve an IncreaseDate Web Service contract and an increasing date specification. Next, we analyze a Web Service contract and an increasing date specification. Subsequently, we generate Web Service operation specification which is used to generate IPM, and then afterwards generate Web Service test cases. However, test suite includes conflict test cases. Therefore, the replace method is conducted in order to remove any conflict test cases in test suite. Table 1 presents the test suite of IncreaseDate service. Figure 13 shows the part of IncreaseDate service's WSTC.

Table 1. The IncreaseDate service test suite.

Test case	Month	Day	Year
1	M1	D1	Y1
2	M1	D2	Y2
3	M1	D3	Y1
4	M1	D4	Y2
5a	**M3**	**D5**	**Y1**
5b	**M1**	**D2**	**Y1**
6	M2	D1	Y2
7	M2	D2	Y1
8	M2	D3	Y2
9	M2	D4	Y2
10	M2	D5	Y2
11	M3	D1	Y1
12	M3	D2	Y2
13	M3	D3	Y1
14	M3	D4	Y1
15	M3	D5	Y2
16	M4	D1	Y1
17	M4	D2	Y2
18	M4	D3	Y1
19a	**M3**	**D4**	**Y2**
19b	**M4**	**D2**	**Y2**
20a	**M3**	**D5**	**Y1**
20b	**M4**	**D2**	**Y1**

IPM is defined as follows [24]:

M1 = {month: month has 30 days}
M2 = {month: month has 31 days except December}
M3 = {month: month is December}
M4 = {month: month is February}
D1 = {day: 1 <= day <= 27}
D2 = {day: day = 28}
D3 = {day: day = 29}
D4 = {day: day = 30}
D5 = {day: day = 31}
Y1 = {year: year is a leap year}
Y2 = {year: year is a common year}

```
<items>
 <item operation=" IncreaseDate">
  <inputs>
   <input>
    <variable>month</variable>
    <value>M1</value>
   </input>
   <input>
    <variable>day</variable>
    <value>d1</value>
   </input>
   <input>
    <variable>year</variable>
    <value>y1</value>
   </input>
  </inputs>
  <results>
   <result>Invoking Web service</result>
   <result></result>
   <result></result>
  </results>
 </item>
 <item operation=" IncreaseDate">
  <inputs>
   <input>
    <variable>month</variable>
    <value>M1</value>
   </input>
   <input>
    <variable>day</variable>
    <value>D2</value>
   </input>
   <input>
    <variable>year</variable>
    <value>y2</value>
   </input>
  </inputs>
  <results>
   <result>Invoking Web service</result>
   <result></result>
   <result></result>
  </results>
 </item>
 …
</items>
```

Figure 13. The WSTC of IncreaseDate service.

As in figure 11 and figure 12, this test suite contains 23 test cases and kills all mutants respectively. In [24], the test suit contains 10 test cases and kills 10 out of 18 mutants. Figure 14 presents IncreaseDate service's code coverage of test cases generated from our approach and decision-base testing.

5. Conclusion and Future Work

In this paper, we apply a pair-wise testing technique to generate Web Service test cases. Web Service operations are identified using WSDL-S and OCL documents. Typically, WSDL-S describes a Web Service contract and OCL expresses service rules including pre- and post- conditions of Web Services. However, some generated test cases may be an impossible combination of two or more IPM. Therefore, this approach will automatically eliminate all conflict test cases with the replace method [21].

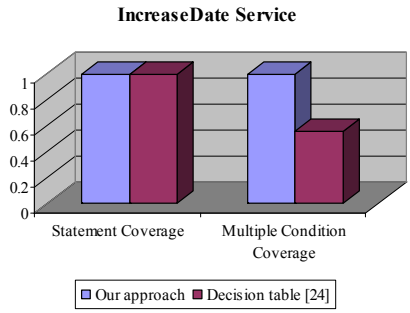

Figure 14. A comparison of IncreaseDate Service' test case coverage.

For future work, we will apply this approach to generate Web Service test case with real-life case study of Web Service and adapt Web Services to using ontology techniques. To suppose the domain ontology defined in a separate OWL specification, WSDL-S can annotate WSDL service specification with the ontology definitions. Hence, the semantic information enables more intelligent analysis.

6. Acknowledgement

This research is part of the Engineering New Paradigm Software for Enterprises with Service-Oriented Architecture Project, supported by Thailand's Software Industry Promotion Agency (Public Organization).

The authors are grateful to Dr. Mala Supongpan and Dr. Robert Jackson for their kindly making through useful comments and also extend their grateful to the Royal Bangkok Sports Club (RBSC) scholarship for partly funded for the project.

7. References

[1] S. Hanna and M. Munro, "Fault-Based Web Services Testing," in *Proceedings of the 5th International Conference on Information Technology: New Generations*, 2008, pp. 471-476.

[2] X. Bai, W. Dong, W.-T. Tsai, and Y. Chen, "WSDL-based automatic test case generation for Web services testing," in *Proceedings of the IEEE*

International Workshop on Service-Oriented System Engineering, 2005, pp. 207-212.

[3] S. Noikajana and T. Suwannasart, "Web Service Test Case Generation Based on Decision Table," in *Proceedings of the 8th International Conference on Quality Software*, 2008, pp. 321-326.

[4] R. Siblini and N. Mansour, "Testing Web services," in *Proceedings of the 3rd ACS/IEEE International Conference on Computer Systems and Applications*, 2005, p. 135.

[5] M. Hong and Z. Lu, "A framework for testing Web services and its supporting tool," in *Proceedings of the IEEE International Workshop Service-Oriented System Engineering*, 2005, pp. 199-206.

[6] W. T. Tsai, R. Paul, Y. Wang, C. Fan, and D. Wang, "Extending WSDL to facilitate Web services testing," in *Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering*, 2002, pp. 171-172.

[7] A. Bertolino, G. Jinghua, E. Marchetti, and A. Polini, "Automatic Test Data Generation for XML Schema-based Partition Testing," in *Proceedings of the 2nd International Workshop on Automation of Software Test*, 2007, p. 4.

[8] S. Noikajana and T. Suwannasart, "An Approach for Web Service Test Case Generation Based on Web Service Semantics," in *Proceedings of the International Conference on Semantic Web and Web Services*, 2008, pp. 171-177.

[9] W. T. Tsai, R. Paul, W. Song, and Z. Cao, "Coyote: an XML-based framework for Web services testing," in *Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering*, 2002, pp. 173-174.

[10] A. Bertolino, J. Gao, E. Marchetti, and A. Polini, "TAXI--A Tool for XML-Based Testing," in *Proceedings of the 29th International Conference on Software Engineering - Companion*, 2007, pp. 53-54.

[11] W3C, "Web Services Description Language (WSDL) 1.1," 2001.

[12] D. Wenli, "Test Case Reduction Technique for BPEL-based Testing," in *Proceedings of the 2008 International Symposium on Electronic Commerce and Security*, 2008, pp. 814-817.

[13] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, Jr., "Software fault interactions and implications for software testing," *the IEEE Transactions on Software Engineering,* vol. 30, pp. 418-421, 2004.

[14] T. O. M. Group, "Object Constraint Language specification," 2006.

[15] J. T. E. Timm and G. C. Gannod, "Specifying Semantic Web Service Compositions using UML and OCL," in *Proceedings of the IEEE International Conference on Web Services, 2007.*, 2007, pp. 521-528.

[16] J. Ni, X. Zhao, and L. Zhu, "A Semantic Web Service-Oriented Model for E-Commerce," in *Proceedings of the International Conference Service Systems and Service Management*, 2007.

[17] S. Battle, S. Williams, and C. Preist, "A Semantic Web Services challenge."

[18] J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam, "WSDL-S: Adding Semantics to WSDL - White Paper."

[19] W3C Member Submission, "Web Service Semantics - WSDL-S," 2005.

[20] R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, A. Sheth, and K. Verma, "Web Service Semantics - WSDL-S."

[21] M. Grindal, J. Offutt, and J. Mellin, "Managing Conflicts When Using Combination Strategies to Test Software," in *Proceedings of the 18th Australian Software Engineering Conference, 2007.*, 2007, pp. 255-264.

[22] F.-a. Qian and J.-h. Jiang, "An Improved Test Case Generation Method of Pair-Wise Testing," in *Proceedings of Asian Test Symposium, 2007. ATS '07. 16th*, 2007, pp. 149-154.

[23] W3C Recommendation, "XML Schema Part 0: Primer Second Edition."

[24] P. Jorgensen, *Software Testing A Craftsman's Approach* 2ed.: CRC PRESS, 2002.