Test Case Reduction Technique for Semantic Based Web Services

A. Askaruinisa
Department of Computer Science and Engineering
Thiagarajar College of Engineering,
Affiliated to Anna University Tirunelveli
Madurai, India

A.M. Abirami
Department of Computer Science and Engineering
Raja College of Engineering,
Affiliated to Anna University Tirunelveli
Madurai, India

***Abstract*—Web Services (WS) are the basic building blocks for every e-business applications. They provide efficient reusability mechanism, thereby reducing the development time and cost. Web services can be identified by Uniform Resource Identifier (URI). The interfaces and bindings of Web Services can be discovered, defined and described as XML artifacts according to Web Service Description Language (WSDL). WSDL can be used to describe web service operations including input, output and exceptions. It cannot identify pre and post conditions of web services. But Semantic WSDL (WSDL-S) identifies the pre and post conditions of web services to generate optimal number of test cases. This paper presents an approach for generating web service test cases using WSDL-S and Object Constraint Language (OCL), while the test case generation technique is Orthogonal Array Testing (OAT). We have developed a prototype namely Semantic Web Services Test Case Generator (SWSTCG) which can be viewed in the web site http://www.tcetesting.webs.com.We have generated WSDL of web service to be tested using NetBeans IDE and converted into WSDL-S by giving OCL references, where pre and post conditions are defined. Test data, using OAT, with different factors, levels and strengths are generated and documented in XML based test files called Web Service Test Specifications (WSTS) and executed. The proposed method is compared with the Pair-Wise Testing (PWT) method. We have conducted testing on various web service applications and the results have shown that the proposed method is effective in generating minimal test cases with maximum test case effectiveness.**

Keywords- web services testing, semantics, test case generation, Orthogonal Testing, Pair-Wise Testing, test case reduction, test case effectiveness

I. INTRODUCTION

Web services are an enabling technique for Service Oriented Computing (SOC) which provides W3C standard based mechanism and open platform for integrating distributed autonomous service components [25]. The quality of services is a key issue for developing service-based software systems and testing is necessary for evaluating the functional correctness, performance and reliability of individual as well as composite services. However, the development of web services is particularly a difficult task due to the complexity of environment in which they must function. One of the most difficult aspects of web service development is the complexity involved in conducting the effective system testing.

The semantic web is an evolving development of the World Wide Web (WWW) in which the meaning (semantics) of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content. The current WSDL standard operates at the syntactic level and lacks the semantic expressivity needed to represent the requirements and capabilities of web services. Also WSDL cannot identify pre and post conditions, logical sequence and constraints of web services. Semantics can improve software reuse and discovery, significantly facilitate composition of web services and enable integration of legacy applications as part of business process integration which can be added through WSDL-S. With respect to web services testing, semantics enable us to reuse test data, thereby reducing the number of test cases to be generated, which in turn reduces test cost and execution time during regression testing.

In order to perform complete testing, Factorial Design Technique or All Combinations Testing (ACT) technique requires more number of test cases. For example, if the application takes 4 parameters and each parameter takes 3 different values, totally we need $3^4 = 81$ test cases. Many combinatorial testing techniques have been evolved to reduce the number of tests. The benefits of combinatorial testing include: dramatically increased test execution efficiency, better quality, better phase containment, increased speed to market and reduced cost of both testing and bug fixing.

Earlier work [1] done on this area considered the PWT technique 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. An N-way testing technique is defined as a set of tests for N parameters, so that every combination of their valid values is covered by at least one test case.

In this paper, we have generated test cases for Web Services using WSDL-S and OCL with OAT technique. We have generated WSDL of Web Service to be tested using NetBeans IDE and converted into WSDL-S by giving OCL references, where pre and post conditions are defined. Test data, using OAT, with different factors, levels and strengths are generated and documented in XML based test files WSTS and executed. The proposed method is compared with the PWT

technique [1]. We have conducted experimental testing on various web service applications and the results have shown that the proposed orthogonal method is effective in generating minimal test cases.

A. Related Work

One of the most difficult aspects of web service development is the effective system testing, which is more complex, as the source code is unavailable to the users of web services. Therefore, much research has been done to improve the web services testing. Many researchers have done their work to generate test cases for web services from WSDL [2, 3, 5, 7, 9, 13] and from WSDL-S using OCL [1, 12]. Xiaoying Bai et al [8] developed ontology based test model for web services using partition testing technique. J Offutt [6] and his team developed test cases for web services using data perturbation. Yongyan Zheng et al and Mounir Lallali et al developed test framework [10, 11] to generate test cases based on finite automata using Business Process Execution Language (BPEL). Timm et al specified [4] semantic web service compositions using Unified Modeling Language (UML) and OCL. In his paper [7], M. Hong considered various fault injection methods for web services and measured the mutation score based on his testing. In [1,12], Sirpol et.al have proposed techniques for generating web service test cases whose work focused on web service contracts based on specifications using WSDL-S and PWT technique.

Reda siblini et al [14] specified testing of web service using mutation analysis. Mutation operators are applied to WSDL document and developed the mutated web service interfaces which are used for testing. Yinong Chen et al [15] provided feedback control model for adaptive testing. The feedback control is used for improving the testing of web services by reducing the cost. In his paper [16], Ashok Kumar dealt about how to automate the web services testing. He proposed a method to parse the WSDL file and generate SOAP requests based upon the parameters (values) from different data base. SOAP requests are then submitted to web server which gives SOAP responses, which are then analyzed. Andre luiz Da Silva Solino et al [17] dealt with mutation testing for web services and compared the results with data perturbation method.

In most of the manufacturing applications, the combinatorial testing technique like OAT has been widely used, but up-to now most of the research work on web service testing is theoretically based on model checking. In this paper we apply OAT technique to minimize the number of test cases for testing the semantic based Web Services. We have generated WSDL of web service to be tested using NetBeans IDE and converted into WSDL-S by giving OCL references, where pre and post conditions are defined. Test data, using OAT, with different factors, levels and strengths are generated and documented in XML based test files WSTS and executed. The proposed approach is compared with existing techniques. We have conducted testing on various web service applications and the results have shown that the proposed method is effective in generating optimal/ minimal test cases.

The remainder of this paper is organized as follows: Section 2 covers the background material required for this proposal, Section 3 describes the approach selected by this paper, Section 4 briefly highlights the implementation methodology used by this proposal and Section 5 gives the conclusion and future enhancements.

II. BACKGROUND

This section introduces the various background materials like WSDL, WSDL-S, OCL and OAT technique.

A. Web Service Description Language (WSDL)

WSDL is an XML-based language to define web services and how to access them. It specifies the location of the service, the operations or methods, and the services exposes.

WSDL Document Structure:

A WSDL document defines a web service using four major elements:

<portType> the operations performed by the web services

<message> the messages used by the service

<types> the data types used by the web service

<binding> the communication protocols used by the Web service.

The main structure of a WSDL document looks like the following:

<definition>

<types>

Definition of types…….

<\types>

<message>

Definition of message…….

<\message>

<portType>

Definition of a port…..

<\portType>

<binding>

Definition of binding…….

<\binding>

<\definition>

A WSDL document can also contain other elements like extension elements and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

The <portType> element is the most important WSDL element. It defines a web service, the operations that can be performed, and the messages that are involved. The

<portType> element can be compared to a function library in a traditional programming language.

The <message> element defines the data elements of an operation. Each message consists of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language.

The <types> element defines the data type that is used by the Web service. For maximum platform neutrality, WSDL uses XML schema syntax to define data types. The <binding> element defines the message format and protocol details for each port.

In this paper, we have generated/retrieved WSDL for the web service to be tested using NetBeans IDE and added semantics to it.

B. Semantic WSDL (WSDL-S)

The web service semantics (WSDL-S) [18, 19] aims to add semantic annotation to web service description by extending WSDL. WSDL-S is an extension of the syntactical level of WSDL and includes semantic capabilities for semantic web services [18]. WSDL-S associates the semantic descriptions to the Web Service in order to enable automatic search, discovery, selection, composition and integration across heterogeneous users and domains. WSDL-S includes three attributes and two elements, in addition to that of WSDL. They are:

- The **precondition** element is a set of assertions that must be met before Web Services can be invoked
- The **effect** element is an element that is a result of invoking a Web Service operation
- The **modelReference** attribute is a specification of association between WSDL entity and a concept
- The **schemaMapping** attribute is a handling structure which differentiates between schema elements of Web Services and their corresponding semantic model concepts
- The **category** attribute is a service categorization of information for publishing a service in a Web Services registry.

This paper uses the **modelReference** attribute to refer the OCL file. We have maintained WSDL-S and OCL files, separately so that future modifications of one would not affect another.

C. Object Constraint Language (OCL)

It is a formal language used to describe expressions on UML models. These expressions typically specify invariant conditions that must hold for the system being modeled or queries over objects described in a model [19]. OCL can be used for a number of different purposes:

• As a query language

• To specify invariants on classes and types in the class model

• To describe pre and post conditions on operations/methods

• To specify target (sets) for messages and actions

• To specify constraints on operations

This paper uses OCL to describe the pre and post conditions of web operations.

D. Orthogonal Array Testing (OAT)

Orthogonal array testing is a systematic and statistical way of testing. Orthogonal arrays could be applied in user interface testing, system testing, regression testing, configuration testing and performance testing. Orthogonal arrays are the extension of Latin Squares [29]. For example, Latin Square of degree 3 is as shown below:

1	2	3
2	3	1
3	1	2

Each value occurs once in each column. Taguchi extended this idea to derive orthogonal array for different factors, levels and strengths [21]. Factors can be mapped to parameters in the application and levels can be mapped to values that each parameter takes. For example, the orthogonal array for factor 3, level 2 and strength 2 is as follows:

1	1	1
1	2	2
2	1	2
2	2	1

Statistical test designs, such as orthogonal arrays, may reduce the number of test cases. Orthogonal arrays combine a set of software parameters into two subsets. One subset may be called the "combination parameters." The second subset may be called the "ancillary parameters." The combination parameters are exhaustively tested. The ancillary parameters are not exhaustively tested [30]. As an example, suppose there are four parameters generically named "A", "B", "C" and "D". Suppose further that each parameter can take on one of 3 values: "1", "2" and "3." With orthogonal array designs, parameters are typically grouped as pairs. In this example, there will be four pairs, "A-B", "B-C", "C-D" and "D-A". Each set of pairs are in turn considered combination parameters to create a set of test cases. Consider the test cases for the "A-B" pair. The combination parameters are "A" and "B," and the ancillary parameters are "C" and "D". Exhaustive testing for "A" and "B," both of which have three possible values, will yield 3^2, or 9, test cases. L_9 (3^4) is the suitable array for this case. Values are assigned to the ancillary parameters, "C" and "D", randomly or based on experience. The process is repeated for the "B-C", "C-D" and "D-A" pairs. The orthogonal design results in 4×9, or 36, test cases. Exhaustive testing would result in 3^4, or 81 test cases.

Orthogonal arrays exhibit the following properties:

Identify applicable sponsor/s here. *(sponsors)*

- Each of the arrays conveys information different from that of any other array in the sequence, i.e., each array conveys unique information therefore avoiding redundancy.
- Each of the arrays is statistically independent of the others.
- Provides uniformly distributed coverage of the test domain.
- Concise test set with fewer test cases is created.
- All pair-wise combinations of test set created.
- Arrives at complex combinations of all the variables.
- Simpler to generate and less error prone than test sets created manually.
- Reduces testing cycle time.

OAT testing method makes sure that each combination is tested *the same number of times*. It also determines which combination break first. In this paper, we have used OAT technique for generating and reducing the test cases.

III. METHODOLOGY

Before The focus of this paper is Web Service test case generation and reduction using semantics and OAT technique.

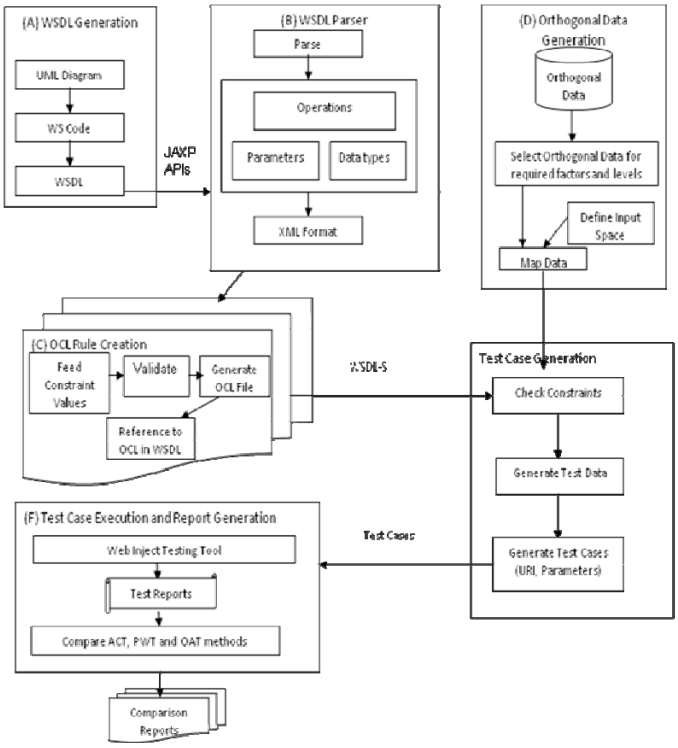

Figure 1 Test Case Generation Framework

In our approach, we describe a Web Service contract using WSDL-S and express pre and post conditions of Web Services using OCL. We have developed a testing framework as in Figure 1 which consists of various modules like (A) WSDL Generation, (B) WSDL Parser, (C) OCL Rule Creation, (D) Orthogonal Data Generation, (E) Test Case Generation and (F) Test Case Execution and Report Generation. The functional description of each module is explained below.

A. *WSDL Generation*

The thought process of application to be executed and tested is represented as UML class diagram as shown in Fig 2 for "Geometry Shape" application. The "Shape" web service consists of an operation createShape() which takes two 'integer' parameters and returns a 'String'. Using the development environment NetBeans IDE, the logic is implemented as web service using AXIS2 [24] plug-in for the interface.

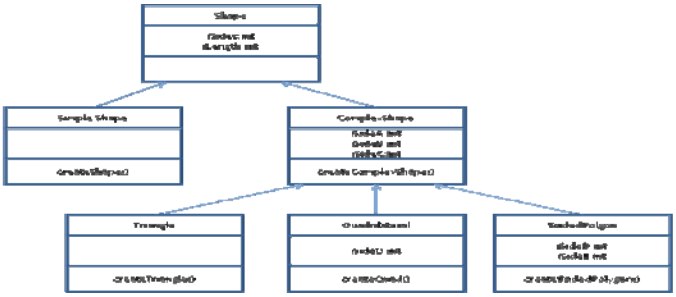

Figure 2 Class Diagram of Shape Service

The IDE generates the appropriate WSDL as in Appendix A. The WSDL thus obtained is given as input to the WSDL parser module.

B. *WSDL Parser*

Since the WSDL generated contains information recursively and quite lengthy, it is parsed using Java XML Processing (JAXP) APIs to retrieve the method signature which will be useful for test case generation. The parameters and their corresponding data types for each operation are determined and stored in the XML format as shown in Fig 3. For example, the operations "createShape" and "complexShape", their parameters and the corresponding data types are parsed and documented in XML format.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
    <opList>
     <opName name="complexShape">
    <parameters name="iSides" type="int" />
    <parameters name="iLength" type="int" />
    </opName>
<opName name="createShape">
<parameters name="iSides" type="int" />
<parameters name="iLength" type="int" />
</opName>
</opList>
```

Figure 3 XML format for Parser output

C. *OCL Rule Creation*

In order to add semantics to WSDL created, we have developed a "Constraint Builder GUI" using NetBeans IDE which feeds constraint or conditions for each parameter of each operation present in the WSDL. All the constraints are built using OCL format and used further for test data selection. The conditions are mentioned by the user/tester. We have kept WSDL-S and OCL files separately, so that any change in the parameter in future may not affect the WSDL file, only the OCL file can be changed. The algorithm to create the OCL file is as follows:

• A service rule name using context is defined.

• Variables name and their data types are defined.

• Each service rule is defined as pre and post conditions using "pre" and "post" tags respectively

• Variables are defined according to data type definitions

The screenshot of the Constraint Builder GUI is shown in Fig 4.

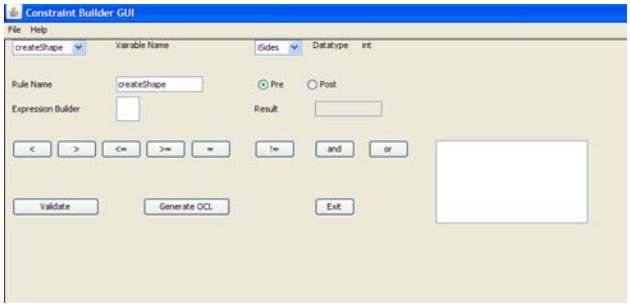

Figure 4 Screenshot of Constraint Builder GUI

The sample OCL file created by the Constraint Builder GUI for the operation createShape() is shown in Fig 5, which describes the pre and post conditions. For example, the parameter "iSides" should be greater than 0 and less than 7 but not equal to 2.

context createShape (iSides:int, iLength:int) : string

pre: iSides > 0 and iSides <7 and iSides != 2 and

iLength > 0 and iLength < 20

post: if (iSides == 1) then result = "Line"

else if (iSides == 3) then result = "Triangle"

else if (iSides == 4) then result = "Quadrilateral"

else if (iSides == 5) then result = "5 sided Ploygon"

else if (iSides == 6) then result = "6 sided Ploygon"

Figure 5 Sample OCL file

The modelReference attribute is added to WSDL to refer the OCL file as shown below:

<wsdl:modelReference="createShape" />

to generate WSDL-S.

The constraints specified in the WSDL-S file are formatted into XML format, as shown in Fig 6, so that the test data can be verified and selected easily for each parameter.

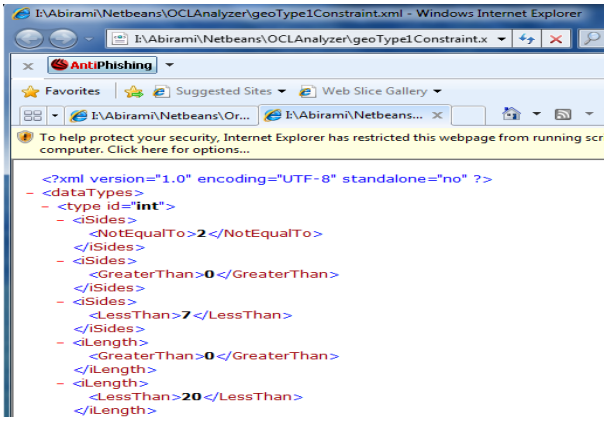

Figure 6 Constraints in XML Format

D. Orthogonal Data Generation

OAT technique is simple and straightforward and can be customized based on available time and cost. The steps to be followed in constructing the orthogonal array are as follows:

- Number of independent variables to be tested is decided, known as the "Factors" of the array.
- Number of values that each independent variable takes is decided, known as the "Levels" of the array.
- A suitable orthogonal array with the smallest number of Runs is selected. Suitable array is one that has atleast as many Factors as needed and has atleast as many levels for each of those factors as decided in.
- Factors and values are mapped onto the array.
- Runs are transcribed into test cases, adding any particularly suspicious combinations that are not generated.

Strength of orthogonal array is the number of columns it takes to see each of the Levels ^{Strength} possibilities occur equally often. Orthogonal arrays are often named using the pattern L_{Runs} (Levels ^{Factors}). In general, strength determines the number of variables to be considered to detect faults i.e the suitable orthogonal array with strength 3 needs to be selected if triple mode fault needs to be detected [27].

If no suitable array is found, then nearly equivalent array has to be selected and discard the unnecessary values or randomly substitute the values [26]. This is explained as follows: Factors {P1, P2, P3, P4} takes values {1,2,3}, {1,2,3}, {1,2,3} and {1,2}. The highest possible level is 3, so the suitable array is $L_9(3^4)$. This is shown below:

Runs	P1	P2	P3	P4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3

5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

But the parameter P4 takes only two values {1, 2}. Hence for the runs #3, #4, #5, the value of the parameter P4 needs to be substituted in a balanced way. In another case, if the array selected based on the number of parameters and levels includes more parameters than are used in the experimental design, ignore the additional parameter columns. For example, if an application has 8 parameters with 2 levels each, the L_{12} array should be selected according to the array selector [26]. The rightmost 3 columns can then be ignored.

Orthogonal combinations of data are obtained as defined by Taguchi [21] by selecting the appropriate factors and levels as shown in Fig 7 and feeding the required input values. In the screenshot shown below, the first parameter takes 6 different values and the second one takes two different values.

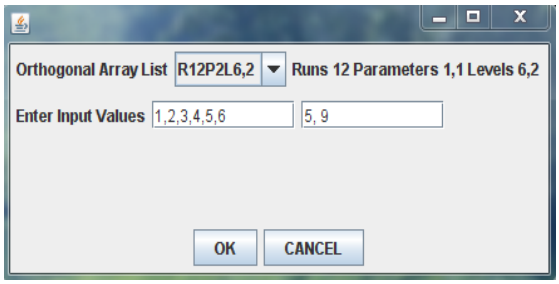

Figure 7 Orthogonal Array Selections

The orthogonal array values are usually kept as the comma separated values in separate flat files. Based on the factors and levels chosen from the above selection GUI, the input values are mapped and the test data is generated in XML format as shown in Fig 8.

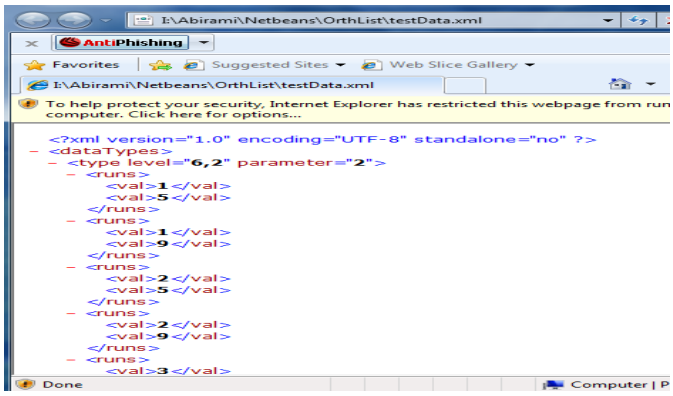

Figure 8 Orthogonal Data Combinations in XML Format

E. Test Case Generation

Based on the test data selection, the test cases are generated in XML format and stored as "testcases.xml" and maintained as WSTS file as shown in Fig 9. Here the tag <case> represents each test case, the attribute "id" represents the test case number, "method" represents get or post method, "url" represents the location of the method, "verifyPositive" represents the expected result value.

```
<testcases>

<case description="Test case for createShape" id="1" method="get"
url="http://localhost:8084/axis2/services/GeoShape/createShape?iSides=1&iLength=5" verifypositive="Line" />

<case description="Test case for createShape" id="2" method="get"
url="http://localhost:8084/axis2/services/GeoShape/createShape?iSides=1&iLength=12" verifypositive="Line" />

<case description="Test case for createShape" id="3" method="get"
url="http://localhost:8084/axis2/services/GeoShape/createShape?iSides=3&iLength=5" verifypositive="Triangle" />

<case description="Test case for createShape" id="4" method="get"
url="http://localhost:8084/axis2/services/GeoShape/createShape?iSides=3&iLength=12" verifypositive="Triangle" />

</testcases>
```

Figure 9 Sample WSTS

F. Test Case Execution and Report Generation

We have considered the WebInject Testing tool [20] for executing the test cases generated. WebInject is the open source testing tool for testing the Web Services, which can be easily installed and used. This tool generates the report both in HTML/XML format. It requires the test cases in the XML format as shown in Fig 9. The WSTS file is given to the WebInject testing tool. Test reports are generated for all combinations of input data. Fig 10 shows the screen shot of test report from the WebInject testing tool [20].

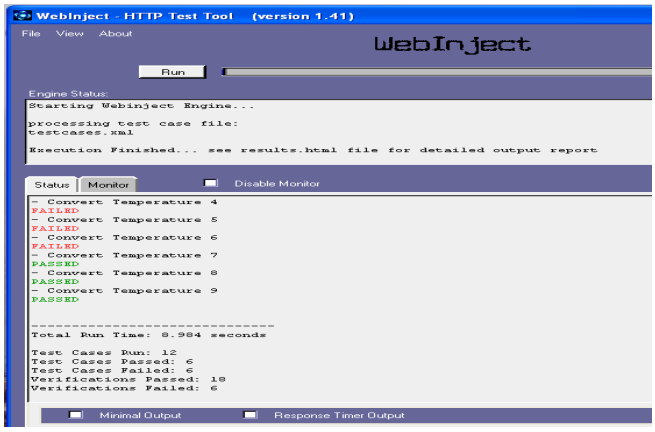

Figure 10 Screen Shot from WebInject Tool

IV. IMPLEMENTATION

We have developed the prototype for generating test cases using JAXP APIs in NetBeans IDE and tested using WebInject

testing tool. We have evaluated the effectiveness of Web Service test cases using OAT method and compared the results with PWT [1] and All Combinations Testing (ACT)/Factorial Design techniques.

A. Subject Applications

We have considered four web service applications namely Shape, ComplexShape, ConvertTemp and ComputerConvert. The first two applications are based on the class diagram of Fig 2 and the other two web services are taken from the web site [23]. The details of these applications are shown in Table 1. For example, the operation "convertTemp()" in the service ConvertTemp takes one "double", two "string" parameters as input and returns "double". Constraints identified for this service are that temperature should be greater than -10, less than 220 and the string values should be from the enumerated values.

B. Data Analysis

For each application, we have parsed the WSDL and created the XML file as shown in Fig 3. We have added constraints to each parameter of web operation as shown in Table 1 using Constraint Builder GUI, thereby converting WSDL into WSDL-S. Then we have obtained the appropriate orthogonal combinations of data and mapped the input values as shown in Fig 7. We have developed the test cases as shown in Fig 9, with the test data matching the constraints specified. The number of test runs required for each application, is as shown in Table 2.

In Table 2, the application Temperature has three parameters where two parameters take two different values {F,C} and one parameter takes 6 different values {-10, 0, 36.9, 98.4, 100, 212}. The different possible combinations(C) for methods PWT [22] and OAT [21] for Factor 3 are tabulated in Table 3.

Applications (.wsdl)	***Tested Web Operations***	***Parameters and Data Types***	***Constraints***
Shape	*createShape*	*(int, int):String*	*iSides > 0* *iSides< 7* *iSides != 2* *iLength > 0* *iLength< 20*
ComplexShape	*createTriangle*	*(int, int, int):String*	*Inherited from Shape*
ConvertTemp	*convertTemp*	*(double, String, String):double*	*1.Temp > -10 and < 220* *2.Input string should be from enumerated value* *3.FromUnit should not be equal to ToUnit*
ComputerConvert	*convertValue*	*(double, String, String):double*	*1.Input should be powers of 2* *2. Input string should be from enumerated value* *3.FromUnit should not be equal to ToUnit*

TABLE I DETAILS OF TESTED APPLICATIONS

TABLE 2 #Test Runs Required for different applications

Application	**Operations**	**Factors**	**Levels**	**Strength**	**# Test Cases**		
					ACT Technique	**PWT Technique**	**OAT Technique**
Shape	createShape	2 (1,1)	2,6	-	12	12	12
ComplexShape	createTriangle	3	2	2	8	4	4
		3 (2,1)	2,6	2	24	12	12
		3 (2,1)	2,10	2	40	20	20
		3 (2,1)	2,14	2	56	28	28
		3 (2,1)	2,22	2	88	44	44
	createQuad	4	3	2	81	12	9
		4	10	2	10000	142	100
	Create5sidedPolygon	5	3	3	243	40	54
		5	4	2	1024	16	16
	Create6sidedPolygon	6	4	3	4096	28	64
		6	5	2	15625	38	25
Temperature	convertTemp	3(2,1)	2,6	2	24	12	12
ComputerBytes	convertValue	3(2,1)	2,6	22	24	12	12

The combinations 3 and 4 are the only valid combinations for PWT technique based on constraints whereas for OAT technique the combinations 4, 5, 6, 10, 11 and 12 are valid. Thus OAT considers more combinations than PWT technique based on the constraints given for the application.

TABLE 3 Input Combinations for Factor 3

C	PWT	OAT	C	PWT	OAT
1	f,f,-10	F,F,-10	7	f,f,98.4	C,C,-10
2	c,c,-10	F,F,0	8	c,c,98.4	C,C,0
3	f,c,0	F,F,36.9	9	f,f,100	C,C,36.9
4	c,f,0	F,C,98.4	10	c,c,100	C,F,98.4
5	f,f,36.9	F,C,100	11	f,f,212	C,F,100
6	c,c,36.9	F,C,212	12	c,c,212	C,F,212

As another example, the five different parameters {A, B, C, D, E} of the operation "create5sidedPolygon" takes different values as follows: the parameters A and B takes {1, 2}, C and D takes {1, 2, 3} and E takes {1,2,3,4,5,6}. The factorial design technique or ACT method needs 2x2x3x3x6=216 test runs. But OAT method needs 49 runs if L_{49} (7^8) array is chosen from the OA catalogue. This can be further reduced to 18 runs if mixed OA $3^6 6^1$ is selected [28]. Mixed Orthogonal Array (mixed OA) is an array which has multiple levels used for different factors. Thus the number of test cases is reduced as the number of parameters increases, when OAT technique is used.

C. Result Analysis

In this paper, we have compared and analyzed our work based on the following criteria.

1. WS Testing with WSDL vs WS Testing with WSDL-S (WSDL vs WSDL-S)
2. Comparison of various test case reduction techniques using WSDL-S

WSDL vs WSDL-S

In this paper, we have compared the number of test runs required for different web services using both WSDL and WSDL-S. For example, in Table 4, for the operation ConvertValue(), nearly 36 test runs are required to test the operation completely. But if the constraints specified are considered for testing, then we need only 12 test runs for the given inputs.

TABLE 4 Comparison of #Test Runs Required for WSDL vs WSDL-S

Operations	Constraints	Input Values	Using WSDL only	Using WSDL-S
createShape	iSides > 0 iSides< 7 iSides != 2 iLength > 0 iLength< 20	iSides = {1,2,3,4,5,6} iLength = {5,12}	12	10
		iSides = {1,2,3,4,5,6} iLength = {5,21}	12	5
createTriangle	Sides > 0, Sides < 20	Sides = { 3, 4}	8	8
		Sides = { -3, 4}	8	0
		SideA = {3,4} SideB = {3,4} SideC = { -3, 4}	8	4
		SideA = {3,4} SideB = {3,4} SideC = {2,4,-5,6,7,8}	24	20
		SideA = {-3,4} SideB = {-3,4} SideC = {2,4,5,6,7,8}	24	6
createQuad	Sides > 0, Sides < 20	Sides = {4,5}	16	16
		Sides = {-4, 5}	16	1
convertTemp	(i) Temp > = -10 and < 220 (ii) FromUnit should not be equal to ToUnit	Temp = {0,100, 98.4, -10} FromUnit= {F,C} ToUnit = {F,C}	16	8
convertValue	**(i) Input should be powers of 2 (ii)FromUnit should not be equal to ToUnit**	**Value = { 64, 1024, 1000, 2000} FromUnit = {Bits, Bytes, Kbytes} ToUnit = {Bits, Bytes, Kbytes}**	**36**	**12**

Thus from the Table 4, it is clear that if semantics are added to WSDL, the number of test cases are reduced saving testing time and effort.

Comparison of various test case reduction techniques

In this paper we have compared and analyzed various test case reduction techniques using WSDL-S as mentioned below considering different criteria like number of test runs generated and various strengths.

1. WS Testing using ACT/Factorial Design Technique and OAT technique
2. WS Testing using ACT/Factorial Design Technique and PWT technique
3. WS Testing using PWT technique vs OAT technique

Comparison of ACT vs OAT

Figure 11 shows the comparison on test runs required between ACT/Factorial Design and OAT techniques. For example, for Factor 4 and Level 10, the ACT method

requires 10000 (10^4) runs but the OAT method requires only 100 runs.

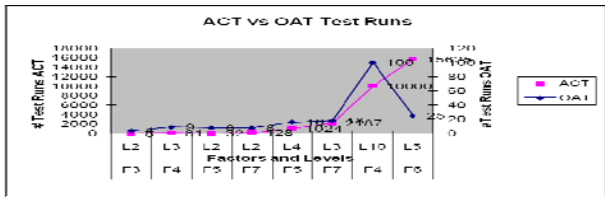

Figure11 ACT vs OAT Test Runs for Various Factors & Levels

Comparison of ACT vs PWT

Figure 12 shows the comparison on test runs required between ACT and PWT techniques. For example, for Factor 4 and Level 10, the ACT method requires 10000 runs but the PWT method requires only 142 runs.

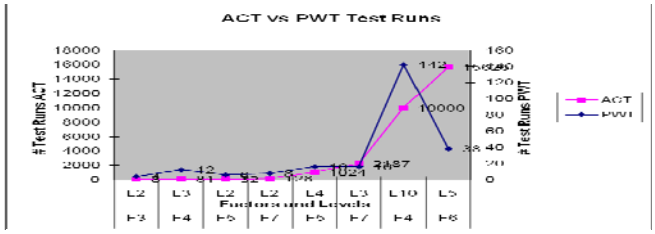

Figure 12 ACT vs PWT Test Runs for Various Factors and Levels

Comparison of PWT vs OAT

Figure 13 shows the test runs required for PWT and OAT techniques for various factors, levels and strength 2. With the increase in parameters and levels, OAT method requires less number of test cases than the PWT. This figure shows that OAT is better technique as the number of parameters increases.

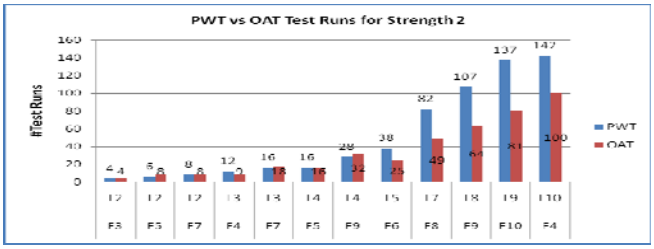

Figure 13 PWT vs OAT Test Runs

But there are cases where OAT generates more number of test runs than PWT technique. For example, in the Figure 13, when the Factor is 5 and Level is 4, OAT generates 18 test cases but PWT generates only 16 test runs.

Comparison on Strengths

Figure 14 shows that if the strength increases, the runs required for OAT increases. Since the OAT method maintains the uniqueness for all parameters using its orthogonal property, the test combinations required are more and it increases with its strength. For OAT method, if the strength considered is 3, then for PWT technique "3-way" is considered. For example, for factor 5, level 3 and strength 3, OAT technique requires 54 test runs, whereas 3-WAY technique requires 40 test runs only as shown in Fig 14.

From these graphs (Figures 11-14), we have proved that, to generate optimal and valid test cases

TABLE 5 #Test Case Effectiveness of PWT and OAT Techniques

Operations	**Factors**	**Levels**	**# Test Cases (TCs)**			**Test Case Effectiveness (in %)**	
			ACT	**PWT**	**OAT**	**PWT**	**OAT**
createShape	2	5	10	10	10	0	0
createTriangle	3	2	8	4	4	50	50
createQuad	4	3	81	12	9	85.18	88.89
create5sidedPolygon	5	4	1024	16	16	98.43	98.43
create5sidedPolygon	5	2	32	8	8	75	75
create6sidedPolygon	6	5	15625	38	25	99.75	99.84

- Combinatorial testing techniques are more effective than factorial design technique.

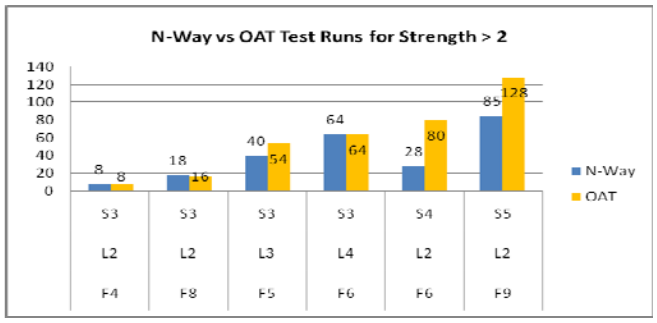

Figure 14 N-Way vs OAT Test Runs for Various Strengths

- As the number of parameters increases, the results show that OAT technique is better than PWT technique.
- As the strength increases, the results show that PWT technique is better than OAT technique.

D. Metrics for Test Case Minimization

The effectiveness of test case minimization is defined using the formula

$$1 - \frac{\text{\# Minimized Test Cases}}{\text{\# Total Test Cases}} * 100$$

Table 5 calculates the effectiveness of test case minimization for the methods PWT and OAT. The test case effectiveness increases with the number of parameters.

As the factors and levels increases, the effectiveness of test case minimization using OAT increases compared to PWT technique. For example, when the Factor is 4 and Level is 3, the effectiveness of test case minimization using OAT technique is 88.89% compared to 85.18% for PWT technique. As the factor increases to 6, the effectiveness of test case minimization using OAT technique increases to 99.84% compared to 99.75% for PWT technique. Thus the reduction technique OAT shows an improvement in testing effort for web services.

V. Conclusion and Future Enhancements

This paper compares the different test case generation and reduction techniques and determines a better technique for testing the web services based on WSDL-S. When there are few parameters, PWT is suitable. When there are more number of parameters (factors) taking different values (levels), OAT is a better technique and also provides reduced number of valid test cases thereby reducing the effort and time.

Our future work will consider the test case prioritization technique for web services based on certain criteria like coverage, cost, quota etc. and to determine a better testing technique by considering the testing efficiency of each test suite.

References

[1]Siripol Noikajana, Taratip Suwannasart, “An Improved Test Case Generation Method for Web Service Testing from WSDL-S and OCL with Pair-wise Testing Technique”, in Proceedings of IEEE International Computer Software and Applications Conference, 2009.
[2] Xiaoying Bai, Wenli Dong, W.-T. Tsai, and Y. Chen, “ WSDL-based automatic test case generation for Web services testing”, in Proceedings of IEEE International Workshop on Service-Oriented System Engineering, 2005, pp.207-212.
[3] Chunyan Ma, Chenglie Du, Tao Zhang1, Fei Hu, “WSDL-Based Automated Test Data Generation for Web Service”, 2008,
[4] John T. E. Timm, Gerald 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.
[5] Sebastien Salva, Isaam Rabhi, “Automatic Web Services Roubustness Testing from WSDL Descriptions”, 2009,
[6] J Offutt, W.Xu, “Generating Test Cases for Web Services using Data Perturbation”, ACM SIGSOFT, Software Eng. Notes, Vol. 29(5), Sep. 2004, pp. 1-10.
[7] 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.
[8] Xiaoying Bai and Shufang Lee Wei-Tek Tsai and Yinong Chen, “Ontology-Based Test Modeling and Partition Testing of Web Services”.
[9] Evan Martin, Suranjana Basu, Tao Xie, “Automated Robustness Testing of Web Services”, 2008.
[10] Yongyan Zheng, Jiong Zhou, Paul Krause, “An Automatic Test Case Generation Framework for Web Services”, 2007.
[11] Mounir Lallali, Faitha Zaidi, Ana Cavalli, Iksoon Hwang, “Automatic Timed Test Case Generation for Web Services Composition”, 2007.
[12] Siripol Noikajana and Taratip Suwannasart, “Web Service Test Case Generation Based on Decision Table”, 2008.
[13] Hanna Samer, Munro Malcolm, “An Approach for Specification-based Test Case Generation for Web Services”, 2007.
[14] Reda siblini, Mashat mansour, “Testing web services”, 2005
[15] Xiaoying bai, Yinong Chen, Zhongkui shao, “Adaptive web services testing”, 31st international computer software and applications conference, 2007
[16] Ashok kumar, “Automated regression suite for testing web services”, 2009 international conference on advances in recent technologies in communication and computing
[17] Andre luiz Da Silva Solino, Sivvia Regina Vergillio, “Mutation based testing of web services”, 2009
[18] “Semantic Web Services Tutorial”, by Michael Stollberg and Armin Haller
[19] “UML 2.0 OCL Specification”, by Object Management Group
[20] http://www.webinject.org
[21] http://www2.research.att.com/~njas/oadir/index.html
[22] www.testerdesk.com
[23] www.webservicex.net
[24] http://netbeans.org/kb/61/websvc/gs-axis.html
[25]www.w3.org/TR/wsdl
[26]http://controls.engin.umich.edu/wiki/index.php/Design_of_experiments_via_taguchi_methods:_orthogonal_arrays
[27] http://www.developsense.com/pairwiseTesting.shtml
[28] http://www.51testing.com/ddimg/uploadsoft/20090113/OATSEN.pdf
[29] http://priorartdatabase.com/IPCOM/000012770/
[30] http://www.faqs.org/patents/app/20090077538

AUTHORS PROFILE

Ms. A. Askarunisa is working in Thiagarajar college of Engineering, Madurai. At present she is pursuing her PhD in Software Testing. She has published papers in National and

International Conferences. Her research interests include Software Engineering, Compilers and Architectures.

Ms. A.M.Abirami is persuing her PG in Computer Science and Engineering. Her area of interests includes testing java applications, web services and web applications.