A Framework for Specification Based Web Service Testing

By
Saif Mohammad Raja Al-Jammal

Supervisor:
Dr. Samer Hanna

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree in Computer Science

Deanship of Academic Research and Graduate Studies
Philadelphia University

June 2014

جامعة فيلادلفيا
نموذج تفويض

أنا سيف محمد رجا الجمال ، أفوض جامعة فيلادلفيا بتزويد نسخ من رسالتي للمكتبات أو المؤسسات أو الهيئات أو الأشخاص عند طلبها.

التوقيع :
التاريخ :

Philadelphia University

Authorization Form

I, Saif Mohammad Raja Al-Jammal, authorize Philadelphia University to supply copies of my thesis to libraries, establishments, or individuals upon request.

Signature:

Date:

A Framework for Specification Based Web Service Testing

By

Saif Mohammad Raja Al-Jammal

Supervisor: Dr. Samer Hanna

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree in Computer Science

Deanship of Academic Research and Graduate Studies

Philadelphia University

June 2014

Successfully defended and approved on _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Examination Committee Signature	**Signature**

Dr, _, Chairman. _ _ _ _ _ _ _ _ _ _ _ _ _
Academic Rank: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dr,_ _, Co-Supervisor. _ _ _ _ _ _ _ _ _ _ _ _ _
Academic Rank: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dr,_ _, Member. _ _ _ _ _ _ _ _ _ _ _ _ _
Academic Rank: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dr,_ _, Member. _ _ _ _ _ _ _ _ _ _ _ _ _
Academic Rank: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dr,_ , External Member. _ _ _ _ _ _ _ _ _ _ _ _ _
Academic Rank: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
( )

Dedication

I fully dedicate this thesis to my family. I dedicate it to my parents, my sons (Mamoon , Ahmad , Mohammad , Rahaf and Aceel) , my wife, my brothers, my sisters and all special friends.

Saif Al-Jammal

Acknowledgment

I would like to express my regards and appreciation to Dr. Samer Hanna who have evaluated my work from the beginning and supported me. I also would like to thank all of those who have helped and encouraged me.

Saif Al-Jammal

Table of Contents

Table of Figures

List of Abbreviations

DB	Data Base
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
J2EE	Java 2 Platform Enterprise Edition
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TLTS	Timed Labeled Transition System
TPG	Task Precedence Graph
TTCN-3	Testing and Test Control Notation
UDDI	Universal Discovery Description And Integration
URI	uniform resource identifier
URLs	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	eXtensible Markup Language

Abstract

The Web Service is a new approach of developing distributed applications which appeared to tackle the differences among program languages and infrastructures. Also, the Web Service interacts the applications with one another whether they are local or remote. The Web Service is a code which is encapsulated by the programmer who is in a remote side. The programmer can use the Web Service by Uniform Resource Locator (URL) and Web Service Description Language (WSDL) which have parameters and their types. As any software, the testing stage must be done. So, many testing techniques appeared to test this kind of application. Every tool concentrated on specific attribute of quality of the Web Service. To use any tool, the tester should have a full understanding of the nature of what the tool can do and how the WSDL could be read then decide whether the specified tool is really suitable for this parameter in the Web Service or not. Depending on the previous explanation, due to the number of the existing, it takes the tester a long time to choose the suitable tool. It also demands that the tester should have enough knowledge about every single tool.

Therefore, this project proposes an approach to select a testing technique automatically. This would reduce the time that the tester uses in selecting the suitable tool as it immediately selects a tool when our approach accesses the Web Service. Our approach selects a suitable testing tool instead of the tester. Thus, the problem of selecting a tool is solved by this approach, and it has the ability to accept a new tool and update the old one if necessary.

The main contribution of this project is to select automatically the suitable testing technique for certain parameter in the Web Service.

The selection is occurred depend on the saved rules of each testing technique. The rules are match with data type and constraining facet in the WSDL file.

CHAPTER ONE

INTRODUCTION

1.1 Introduction

The widespread use of Web Services in commercial applications requires the adoption of developed techniques to ensure the quality of Web Services. Software testing plays an important role in assessing the quality of Software applications. However, existing testing techniques are much older than the Web Services technology and are not appropriate to be used with Web Services (A. de Melo and P. Silveira, 2011). For this reason, the current testing techniques and tools must be modified in order to make it useful to be used to assess Web Services quality attributes.

There are many testing techniques, equivalence partitioning, and boundary value. The tester could be confused with select which proper testing technique for the data type of specific parameter. This project proposes approach to select automatically the testing technique instead of the user. For simple case, this project uses some of techniques such as the equivalence partitioning and boundary value (C .Hart et al, 2005). The first one is black box testing technique which is used to divide input domain to various parts. It is used to reduce the number of test cases. So, the testing some valid cases can dispose remaining elements while the second one is a technique which focuses at edges of intervals. The selected values are seven in general such as min-1, min, min+1, nominal, max-1, max, max+1. In general, most errors occur at the edges. So, this technique detects the errors at the boundary rather than the middle of interval. Our target is to build an approach that automatically selects a testing technique which tests suitable parameters of Web Service.

This introduction section will give a definition for Web Services (Section 1.2), and after that, Web Services testing and specifically Web Services modified the value of testing will be explored (Section 1.3).

1.2 Web Service and WSDL

Web Service is an application that supports application integration and interoperability of systems through network applications. Web Services allow applications to access data that was difficult to reach over heterogeneous networks and exchanging information in a simple, standardized manner.

Following is the definition of World Wide Web Consortium (W3C); Web Service is (P. Ammann and J .Offutt, 2008):

"A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Services in a manner prescribed by its description using Simple Object Access Protocol (SOAP), typically conveyed using Hypertext Transfer Protocol (HTTP) with eXtensible Markup Language (XML) serialization in conjunction with other Web-related standards".

Web Services are based on the Internet and open standards such as eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) (A. Lyer, 2009).

XML format which is used by Web Service enable services to interact with each other over heterogeneous networks, for instance: Microsoft .NET, Sun J2EE, run on Window or Linux, can communicate using a common format (XML format). XML Schema (B. Stepien and I. Schieferdecker, 2003) provides a way to define the data types, and SOAP specifies how to encode an HTTP header and an XML file so that a program in one computer can call a program in another computer and pass it

information. It also specifies how the called program can return a response. An Internet protocol as HTTP is usually used to exchange these XML-based messages from/to other services.

A Web Service must be advertising its invocation interface - its specification and functionality description - this can be done using Web Services Description Language (WSDL) standard. Then services are registered in a Universal Discovery Description and Integration registry (UDDI), which allows each Web Service to be discovered by other services.

Web Service is one way to implement the Service Oriented Architecture (SOA) - the ways of building software applications in SOA are services.

WSDL plays a significant role in the overall Web Services architecture since it describes the complete contract for application communication.

A WSDL file contains all of the information necessary for a client to invoke the methods of a Web Service including:

- The URLs used to access the Web Service.
- How the operation is invoked.
- Definition of one or more services (what a Web Service can do).
- The protocols' and messages' formats allowed for each method
- The input and output messages used by an operation as a method parameters or return values.
- Any complex data types used in the WSDL document.

- And the operations that can be performed by a Web Service, what operations are available on the server.

1.3 Web Service Testing

Software Testing is a Software Engineering technique that is aimed at examining a quality attribute or the capability of a program or system and determining if it meets its required specifications or not. Thus, it is an important technique for assessing the quality of a software product. The purpose of testing is to detect the vulnerabilities in a system such as not being able to handle an invalid input.

Research's results found that software organizations are spending up to 40% of their resources on testing (D. Booth et al, 2004).

As with traditional systems, Web Services must be tested at the unit and integration levels.

Testing SOAP messages addresses request/response mechanisms and data format aspects of Web Services. WSDL is used to expose interfaces as services available on the Internet. Testing WSDL files can be used to generate test plans to validate services. Testing UDDI registries provides the capabilities of publishing, finding and binding of SOA, giving the way software is integrated.

The present work focus mainly on data perturbation testing techniques for SOAP messages (A. de Melo and P. Silveira, 2011).

The quality attribute of a Web Service will be tested here using new software that selects proper technique which matches with the type of attribute.

1.4 Problem statement

The previous research related to Web Services data perturbation testing such as (P. Ammann and J. Offutt, 2008) and (A. de Melo and P. Silveira, 2011) considered perturbing the data types of the Web Services input messages parameters data types by considering, mainly, the boundary values of the parameters, however, other perturbation approaches can be used to test Web Service such as applying the syntax based perturbation of the input parameters data types.

the tester should have a full understanding of the nature of what the tool can do and how the WSDL could be read then he decide whether the specified tool is really suitable for this parameter in the Web Service or not. Depending on the previous explanation, due to the number of the existing technique, it takes the tester a long time to choose the suitable tool. It also demands that the tester should have enough knowledge about every single tool.

Accordingly, the main problem to be considered in our project is to introduce an approach that automatically selects testing technique to test Web Services. According to the constraining facets in the WSDL and depending on the technique itself the data can be generated.

1.5 Motivation

Our motivations are:

1. Allowing the service requester to choose the appropriate testing technique automatically
2. Allowing the service requester reaching high speed in testing Web Services.
3. Providing inexperienced clients with a mechanism to test Web Services regardless of their testing background.

1.6 Major contribution and objective

Introducing an approach to test Web Services that is:

1. Extensible techniques, the approach can accept any new testing technique such as perturbation testing, robustness testing etc.
2. Doesn't need any experience in testing Web Services.
3. Reduce time taken in the testing process. The user should do many steps to start the testing, and generate test cases. These steps will be done automatically by our approach.

1.7 Organization of the Project

This project is divided into six chapters:

Chapter 2: Background

In this chapter, the state of the art in Web Service, WSDL, SOAP, UDDI, XML, and XML schema are given. In addition to that, necessary background for black box testing, boundary value analysis, Equivalence partitioning, and Syntax test are presented and discussed.

Chapter 3: Related Work

In this chapter, the relative works of Web Service are shown that Web Service can be used anywhere. Many testing techniques are presented.

Chapter 4: CHOOSING A SUITABLE TEST TECHNIQUE

In this chapter, the automatic selected testing technique of Web Service is presented. The ability to insert and drop testing technique is explained.

Chapter 5: Evaluation

.In this chapter, since to the best of our knowledge this approach is the first in this field. So, the example is shown its quality.

Chapter 6: Conclusion and Future Work

This chapter discusses and reviews the achieved goals and their efficiency in selecting testing technique, as well as presenting future research problems are.

CHAPTER TWO

BACKGROUND

2.1 Introduction

This chapter covers the fundamentals of Web Service which tackles heterogeneous computer environments and some testing techniques which are applied to ensure the quality of Web Services.

2.2 What is the Web Service?

The rapid users' demands are increasing with the change of the demands of business. To meet the users' needs, the development in the programming language started in the last decade. The idea of programming gives a specific service to users. The programmer deals with local code and perhaps uses a remote one. When the service is ready, it usually has a GUI. By using the internet, Web sites are existed. The web site consists of images, texts, and other media that interact with the user. The web site has many web pages that are connected by using hyperlinks. The web site may have a Web Service. The Web Service isn't a site by itself but it is found within the web sites. The Web Service is a method that is consumed remotely or locally. So, the user passes parameters to the Web Service. These parameters could be simple or complex. Undoubtedly, the Web Service must responds to the request of the user. The user may be another Web Service, program, or a user.

To deal with Web Service in a good manner, you should have a solid knowledge about some techniques. These techniques are WSDL, UDDI, and SOAP protocols which are used to interact with the roles of the Web Service.

2.2.1 The Web Service Model

The Web Service model consists of three parts. These parts are the service provider, service requester, and service broker as shown in figure (2-1).

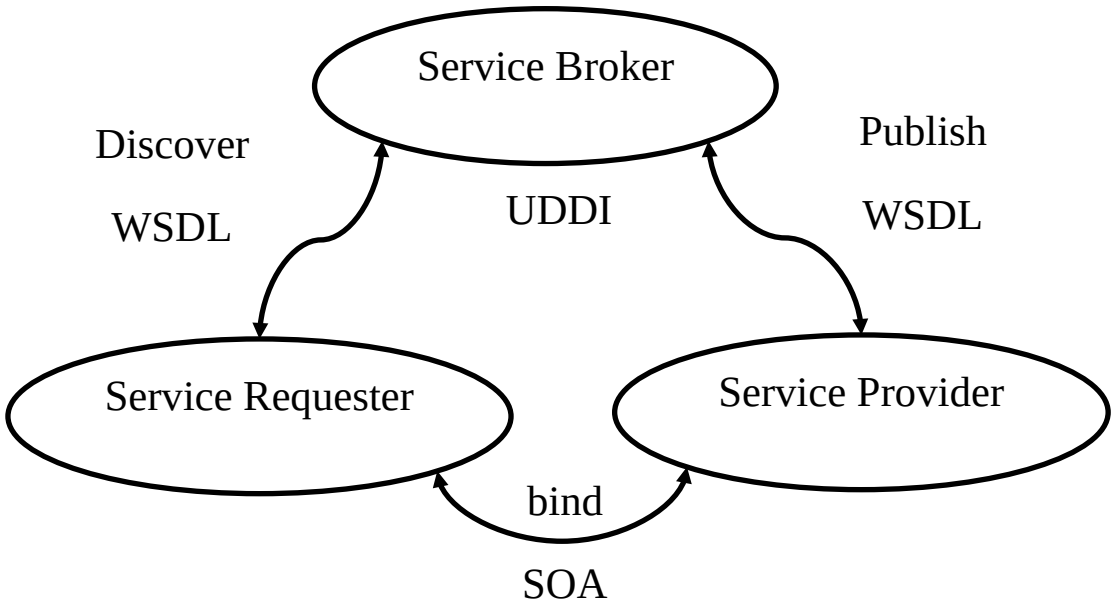

Figure 2-1 Web Service Model.

The responsibility of the service requester is to discover the Web Service while the service provider is the owner of the Web Service, and the service broker is used as a registration of where the service provider is.

2.2.2 The Web Service Protocol

The following are the protocols that are used in the Web Service model to interact through the model:

2.2.2.1 WSDL

Web Service Description Language is a structure that is used to describe the Web Service through the internet. The operations and arguments are available through it. The metadata are used on the client's side to consume the Web Service. The WSDL is XML-based, and it is a machine readable format. However, it can be read regardless of infrastructure. The WSDL structure consists of some abstract elements as follows:

Types: which describe the type of xml schema.

Message: it is not as the message sent from the requester to web server which require service. It is a logical description of the data in the message between two sides, Web Service on the server side and the client on the user side.

Operation: is an abstract definition of method in the Web Service.

PortType: is a set of Operations. Since the WSDL can defines four types such as one-way, Request-response, Solicit-response, and Notification.

2.2.2.2 SOAP

The Simple Object Access Protocol is a standard used to send and receive messages over the internet. The http is responsible for requesting and responding to the message. The SOAP is an XML based over the network. The figure (2-2) shows an example of SOAP message. The message has many elements. These elements construct the general structure of the message. In the body element, the value of age is 18 as instance which is sent to the web service. I.e the body has the values to be sent. This is general structure which is applied in both local and remote web service. The figure (2-3) represents the general structure of remote real web service, url:"

http://www.webservicex.net/CurrencyConvertor.asmx?op=ConversionRate". The figure (2-3) shows the request and response. The figure (2-4) shows instance of remote real web service response.

```
<!-- Request document -->
<soapenv:Envelope
xmlns:xsi="http://localhost/2001/XMLSchema-instance"
xmlns:xsd="http://localhost/2001/XMLSchema"
xmlns:soapenv="http://localhost/soap/envelope/">
<soapenv:Body>
<invoice xmlns="http://localhost/study ">
<age>18 </age>
</soapenv:Body>
</soapenv:Envelope>
```

Figure 2-2 An Example Local of SOAP message

```
POST /CurrencyConvertor.asmx HTTP/1.1
Host: www.webservicex.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.webserviceX.NET/ConversionRate"


<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ConversionRate xmlns="http://www.webserviceX.NET/">
      <FromCurrency> EUR or JOD or USD </FromCurrency>
```

```
    <ToCurrency> EUR or JOD or USD </ToCurrency>
   </ConversionRate>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ConversionRateResponse xmlns="http://www.webserviceX.NET/">
      <ConversionRateResult>double</ConversionRateResult>
    </ConversionRateResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 2-3 An Example remote of SOAP message

```
<?xml version="1.0" encoding="UTF-8"?>
<double xmlns="http://www.webserviceX.NET/">0.7082</double>
```

Figure 2-4 An Example of instance for SOAP message

2.2.2.3 UDDI

Universal Description, Discovery and Integration is the directory where the Web Services are located. There are three main vendors to discover the Web Services: IBM, Microsoft, and HP. Through UDDI, the application of the .net can consume the Web Service and deal with it easily.

2.3 XML

XML stands for eXtensible Markup Language. This language is used to represent data and interchange them in a web-based application. The XML represents data in a hierarchical manner. It uses logical tags related to the name of data. The symbols '<' and '>' are placed around tags. Figure (2-3) shows a simple instance of XML applied for grading students. In this instance, Grade is tag name, <Grade> is start tag, </Grade> is end tag, and 85 is element content.

```
<Student>
        <Grade>
                85
        </Grade>
</Student>
```

Figure 2-5 simple instance of XML

2.4 XML Schema

It is a description of XML document and it is written in an XML document too. The schema for XML defines elements, attributes, and rules among other items. The primary components in schema are: simple type, complex type, attribute, and element. The secondary components are attribute group, identity-constraint, model group, and notation. The helper components are annotations, model group, particles, wildcards, and attribute uses.

Figure (2-4) shows XML schema with constraint facet for XML that has restriction to grade.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="Student">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Grade">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:integer">
                     <xsd:minExclusive value="80"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Figure 2-6 XML schema with restriction

Some of the primary components will be described in more depth in the following subtitles

2.4.1 Data Types:

XML schema has many types to represent the values. These types are named as other programming languages. Many of these data types are: (Any URI, Base64Binary, Boolean, Byte, Date, dateTime, double, float, Duration, gDay, gMonth, gMonthDay, gYear, gYearMonth, hexBinary, Integer, String, Time)

2.4.2 Constraining Facets

The XML schema is used to describe the XML document. The constraint facet in the XML schema is used to restrict the values parameters. There are many constraint facets where each one applies to many types of data. The overlap between constraint facets is acceptable where the same data type can be used within more than one

constraint facet. Many data types accept many constraint facets i.e not only one constraint facet for each one. The following constraint facets are: (enumeration, fractionDigits, length, minExclusive, maxExclusive, minInclusive, maxInclusive, maxLength, minLength, pattern, totalDigits, and whiteSpaces).

2.5 Black-box testing

The testing technique is used in the system development life cycle. Testing has two types: Black-box and White-box. The black-box deals with input and output values of the function. The white-box deals with the implementation of the system. In this section, three testing technique with type of black-box are explained and used in this project.

2.5.1 The boundary value analysis

This testing technique is black-box type. This test does not only focus on the edge of intervals, but it also focuses on the input variables of the method. The reason of doing these tests is the number of errors that occur on the edges of the domain. On the other hand, to test input variables, the tool should not test all possible values that are logically accepted in this variable.

2.5.2 Equivalence partitioning

This testing technique is black box type. This test assumes that there are many partitions of data. The same condition can be applied to each part of a partition without any modification in the condition. So, if the condition cannot be applied to at least one part of group, the condition will not be applied to the rest of the group.

2.5.3 Syntax test

This type of test is black-box. It is specified to string data type. In Web Service, it is specified to the enumeration which, for sure, includes the string as mentioned in the (S. Hanna and M. Munro, 2009). Syntax relies on the structure of words. Norms are applied syntax for generalized it. So, if the word follows the roles of grammar, it is valid; otherwise, it is invalid. For instance,

2.6 Summary

This chapter explains the fundamentals of the Web Services. Using Web Service appeared to process the heterogeneous programs. The black-box testing technique explained with examples to be used later in this project.

CHAPTER THREE

RELATED WORK

3.1 Introduction

This chapter, an overview of some references which deals with testing is presented.

3.2 Testing the Web Service

The authors in (M.T. Girish and R.R Mudholkar, 2013) proposed a framework to test the Web Service. This framework helps the tester of the Web Service by giving a guide in double ways to run the test data for the testing before acquiring effective test data. After that the author debates some views for the implementation then gives a description on the experiment about the framework.

The authors in (C. Mainka et al, 2012) motivated a WS-Attacker automated penetration test tool and described its fundamental to analyze the security of XML interfaces. According to the penetrations the result has triple result, success or not, it produces a kind of log entries that can be filtered by their importance level and may contain additional info's.

The authors in (S. Salva and I. Rabhi, 2009) proposed a robustness testing method that automatically create and run test cases. These cases are created from WSDL descriptions. The authors analyzed and improved the Web Service observability and robustness respectively. They tried to reduce the test cost. The robustness is the ability for the web service not to hang or crash if unexpected event is occurred. The observability is the number of occurred events.

The writers in the (C. Bartolini et al, 2008) paper presented a combination between two tools, SOAP UI and TAXI (C. Pik Wah, 2008). They integrate the operations and data-driven test cases generation with each other. The TAXI derive instance of XML from an XML Schema automatically. This work derives a test message from WSDL

automatically. The test message is valid in the Web Service and it should be able to handle. (Delete it).

The authors in this paper (J. Garc´ıa-Fanjul et al, 2006) proposed a method for testing the composition in the Web Services. This tool is a formal verification and it is used to generate test suites automatically for the composition. A specific condition is there to define a systematic procedure.

The authors in this paper (X. Bai et al, 2005) show generation test cases from WSDL file. The operation separated into two stages parsed and transformed the WSDL file then generated test cases with their perspectives.

The authors (W.T. Tsai et al, 2005) show that the testing is improved from basic two-phase to efficiency testing one. In the same time, the huge numbers of Web Services are separated into small window then the next window.

The authors in (N. Mansour et al, 2005) presented a technique that combined many Web Services to build a reliable web application. There is a guarantee for the availability of proper Web Services at invocation time. The Task Precedence Graph (TPG) which is a web application and Timed Labeled Transition System (TLTS) which is the behavior of the combined components is the two-level abstracted model that specified web application and its combined components. The WSDL file, the TPG, and the TLTS are used to generate sets of test cases. Its emphasis that web service play good role to build web application and another usage WSDL used as assistance in test stage.

The authors in the paper (B. Stepien and I. Schieferdecker, 2003), view elastic testing framework for the Web Service. The framework maps of XML to Testing and Test

Control Notation TTCN-3. The test is provided by the framework loads tests, service interaction, and functions. The mapping enables the automated derivation of test data.

In this paper, (W.T. Tsai et al, 2002) presents a specification based robust testing framework for Web Services. The authors mention the angles that the developer must take into consideration in testing operation. Some of the factors are QoS and interoperability of the Web Service.

The author in (J. Bloomberg, 2002) shows three phases of the history of the Web Services testing. These phases are classified based on their functionality. Phase one extends from 2002 to 2003. In this phase, the most important capabilities of the Web Service testing tools are Testing SOAP message, Testing WSDL files and using them for test plan generation, and Web Service consumer and producer emulation. The second phase extends from 2003 to 2004. Testing publishing, finding, and binding capabilities of the Web Services are done in this phase. The last phase, Testing Dynamic Runtime Capabilities is done in 2004 and after.

Since there are many techniques to test the Web Service, this project proposed a new approach to decide which technique is suitable for a specific Web Service. Surely, the selection is automatic and it depends on the XML schema which is extracted from a WSDL file.

3.3 Summary

The many of research is done in the field of testing the Web Service. Since the Web Service is used widely, the Web Service is located in a remote server. The Web Services interact with each other using messages. As previous researches show, there are many techniques used in testing the Web Service. The stages from the server to the client are tested using various approaches. This variety motivates this project to make a framework that selects the previous techniques automatically.

CHAPTER FOUR

APPROACH TO CHOOSING A SUITABLE TEST TECHNIQUE AUTOMATICLY

4.1 Introduction

After accessing the Web Service, we can get the description of that service. Through that description, we can get the XML schema. The XML schema contains the description of the XML file which presents the message between the client and the server. Sometimes, there are restrictions on the data. This message suggests an automatic selection to the suitable technique for checking reliability of the Web Service.

There are many constraint facets in the XML Schema, enumeration, fractionDigits, length, minExclusive, maxExclusive, minInclusive, maxInclusive, maxLength, minLength, pattern, totalDigits, and whiteSpaces.

The enumeration is a group of constant values that have special formation. The fractionDigits is attention in fraction part with specified number of digits. The length is the number of X where X depends on data type which the length is beyond to. The minEnclusive value includes the lowest announced value but the minEnclusive value is not included. The maxEnclusive includes the highest announced value but the maxEnclusive value is not included. The minInclusive value includes the lowest announced value. The MaxInclusive includes the highest announced value. The maxLength is maximum allowable length the value can accept. The minLength is minimum allowable length the value can accept. The pattern is general structure that restricts the format of expression. The totalDigits are number of digit real and fraction in the value. The whiteSpaces is special character such as tab in the string.

4.2 The Model

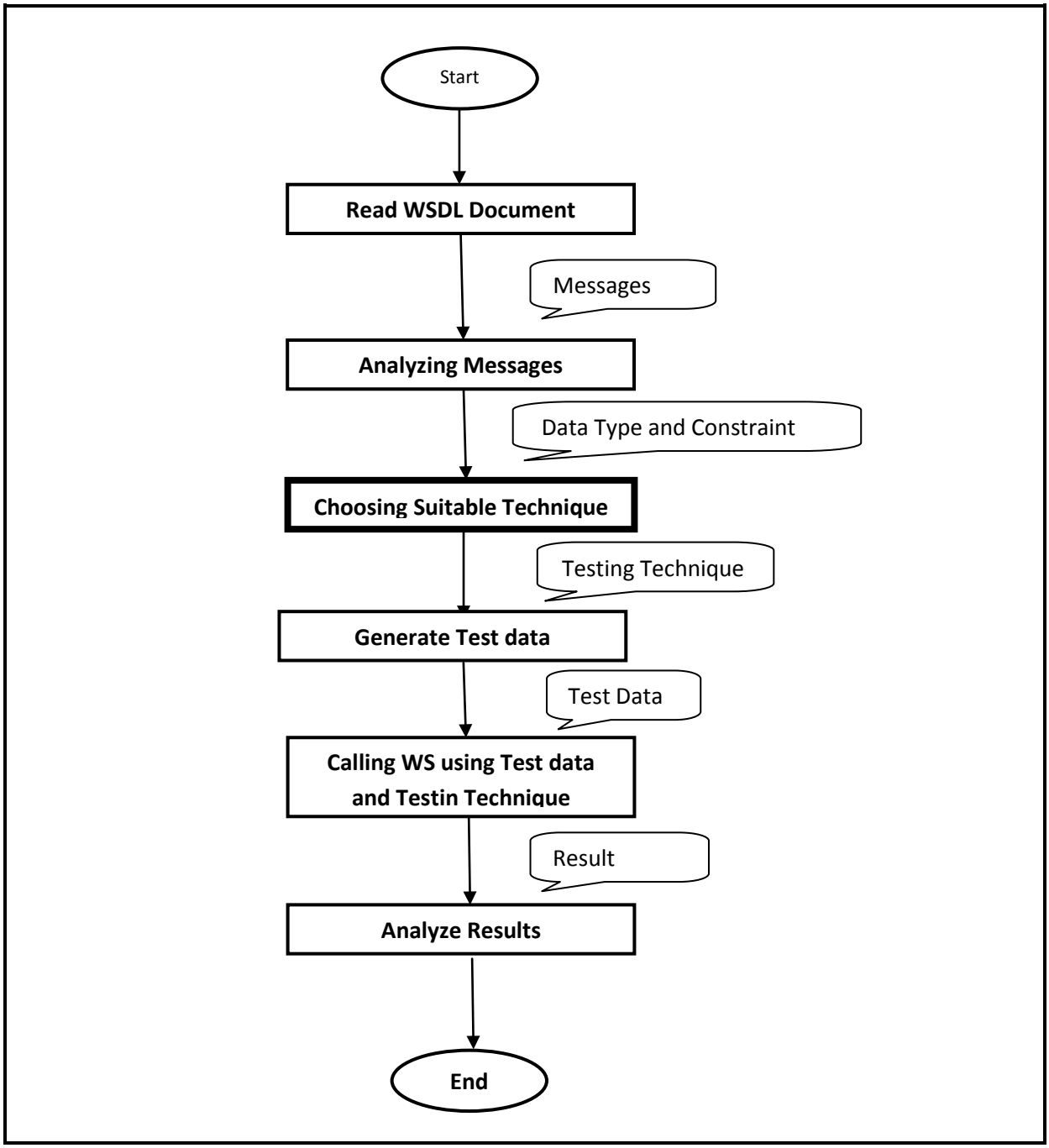

Figure 4-1 The Model of Suitable Technique

Figure (4-1) is illustrating the model of our project. As shown the model consists of five parts: Read WSDL document, analyzing messages, choosing suitable technique, generate Test data, and save to xml file. These parts are explained next in more details:

Reads WSDL document: this part is responsible of reading the description file (WSDL) of the web service under test. Our approach accesses the web service through its link then it gets the WSDL file by adds (? WSDL) to the link or uses special objects response to read WSDL element by element until access the message block, after that send this messages to next stage .

Analysis messages: received the extracted messages and detects the data types and constraining facet then send the data types and constraining facet to next stage.

Choosing suitable technique: is the most significant part in our approach. The number of testing techniques confuses the tester regarding the proper technique that should be used on specific parameter of the Web Service. Since each technique has its own methodology, the tester must firstly understand the methodologies then know each technique's usage and then select the best one.

Our model automates the process of choosing the best available testing technique as shown figure (4-1).

Figure (4-2) presents the framework of choosing testing technique based on input parameter data type and constraining facet.

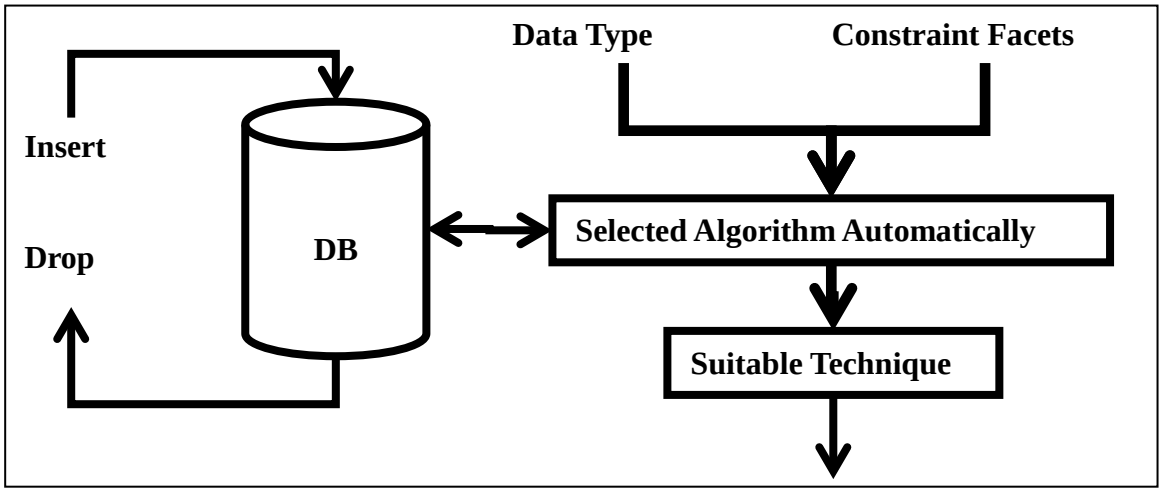

Figure 4-2 Choosing Suitable Testing Technique Framework

The framework has a data base (DB) to store the testing techniques. There is an ability to insert – delete a testing technique, it also updates an existing one if needed. The selected technique process accepts two parameters, input parameter data type and constraining facet. Finally the framework sends the testing technique to another stage to generate testing data.

The selection method is based on specific rules. These rules depend on data type and constraining facet that are extracted from the WSDL of the Web Service under test. The general rules of choosing testing technique with specific data types and constraining facets are as follow:

1. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet = { (minInclusive and maxInclusive) } then technique is boundary value .
2. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet ={ (minEnclusive and maxEnclusive) } then technique is boundary value
3. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or PositiveInteger or Short or unsignedLongInteger or unsignedInt or unsignedShort or unsignedByte) and constraint facet = {fractionDigits or totalDigits } then technique is boundary value
4. if data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = { minLength and maxLenght } then technique is boundary value .
5. if data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = { Length } then technique is boundary value .
6. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or

unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet = * { (minInclusive and maxInclusive) } at union then technique is equivalence.

7. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet = * { (minEnclusive and maxEnclusive) } at union then technique is equivalence.
8. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or PositiveInteger or Short or unsignedLongInteger or unsignedInt or unsignedShort or unsignedByte) and constraint facet = * (fractionDigits or totalDigits) at union then technique is equivalence.
9. if data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = * { minlength and maxlenght } at union then technique is equivalence.
10. if data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = * { length } at union then technique is equivalence.
11. if data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = {pattern or white space or enumeration} then technique is syntax test .

As prove to this rules the authors in (S. Hanna and M. Munro, 2009) used techniques according to the fixed conditions in the above rules (from 1 to 11). So, the completeness of the above rules depends tightly on the completeness in these research works.

When our approach determines the suitable technique, it generates test data using the following test data generation rules:

a. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet = {(minInclusive and maxInclusive)} then test data={ Min-1,Min,Min+1,nominal , Max-1,Max,Max+1}

b. if data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet ={(minEnclusive and maxEnclusive)} then test data ={ Min-1,Min,Min+1,nominal , Max-1,Max,Max+1}

c. If data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet = minInclusive or minEnclusive then test data = {Min-1,Min,Min+1}

d. If data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or Short or unsignedLong or unsignedInt or unsignedShort or unsignedByte or Double or Float or date or DateTime or Duration or gDay or gMonth or gMonthDay or gYear or gYearMonth or Time) and constraint facet = maxInclusive or maxEnclusive then test data ={Max-1,Max,Max+1}

e. If data type = (Byte or Decimal or Int or Integer or Long or negativeInteger or nonNegativeInteger or nonPositiveInteger or PositiveInteger or Short or unsignedLongInteger or unsignedInt or unsignedShort or unsignedByte) and

constraint facet = fractionDigits or totalDigits then test data = {the number of digits is less or more the value of fractuinDigits or totalDigits, and the same }

f. if data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = { minLength and maxLenght } then test data = { string with MinL-1,string with the exact MinL, string with MinL+1 , nominal of Length, string with MaxL-1,string with the exact MaxL, string with MaxL +1}

g. If data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = Length then test data = {length-1,length,length+1}

h. If data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = enumeration then test data = {value not included in set,value in the set,null}

i. If data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = whiteSpace then test data = {null, tab, space,line feed }

j. If data type = (String or Entity or ID or IDREF or Language or Name or NCName or NMTOKEN or NormalizedString or Notation or Token or AnyURI or QName or Base64Binary or HexBinary or ENTITIES or IDREFS or NMTOKENS) and constraint facet = pattern then test data = {the value deduced from the pattern, apply the same format with modified values which protect the general structure }

Where:

*: have many

The generated test data using the pervious rules are saved to an xml file to be used in the next stage to test the functionality of web service.

Calling WS using Test data and Testing Technique: this stage uses the test data and testing technique that was generated from previous stages to consume the Web Service. So, this stage builds SOAP messages which include test cases that are generated previously. After that send the results to the next stage.

Analyze Results: This part receives the response SOAP message from the Web Service. The response may have expected returned value so in this case the perturbation data is valid but if the returned value is arbitrary or unexpected then sent data is invalid. The conclusion of this stage may valid or invalid for the quality of Web Service.

The following figure (4-3) is simple examples of xml schema extracted from WSDL that deal with three techniques:

```
<xsd:simpleType name="C">
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="70"/>
            <xsd:maxInclusive value="79"/>
        </xsd:restriction>
  </xsd:simpleType>
```

(A) Example for Boundary Value schema

```
<xsd:element name="Student" >
  <xsd:complexType>
        <xsd:sequence>
           <xsd:element name="Grade">
              <xsd:simpleType>
                   <xsd:union memberTypes = "C B"/>
              </xsd:simpleType>
           </xsd:element>
        </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
<xsd:simpleType name="C">
    <xsd:restriction base="xsd:integer">
         <xsd:minInclusive value="70"/>
         <xsd:maxInclusive value="79"/>
     </xsd:restriction>
 </xsd:simpleType>
 <xsd:simpleType name="B">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="80"/>
      <xsd:maxInclusive value="89"/>
    </xsd:restriction>
</xsd:simpleType>
```

(B) Example for equivalence partitioning schema

```
<xsd:simpleType name="First_Name">
  <xsd:restriction base="xsd:string">
       <xsd:enumeration value='Join' />
       <xsd:enumeration value='Fara' />
  </xsd:restriction>
</xsd:simpleType>
```

(C) Example for Syntax test schema

Figure 4-3 Simple example for techniques extracted from WSDL

4.3 Summary

Testing the Web Service is a very important stage for the vendors to publish their Web Services to be in commercial usage. Since there are many testing techniques, the tester may be confused to select the proper technique to test the specific data type. This chapter demonstrated automatic selection approach to select the suitable technique to test the input parameter for the Web Service regardless of the knowledge of the user.

CHAPTER FIVE

IMPLEMENTATION AND EVALUATION

5.1 Introduction

This chapter gives us the effect of applying our approach on the Web Service. The Web Service is a technology that is used to make the applications interact with each other. The Web Service can run in different environments.

5.2 Used Environment

The environment consists of server side and client sides. This research builds both server and client on the same machine which has the following specifications:

Platform: Windows 7 Professional.

System: Microsoft Visual Studio 2010

5.3 Evaluation

We analyze more than 100 WSDL file and experiment the correctness of our approach using these WSDL'S. As a case study we apply our approach on a real Web Service chose the from selected sample which we discus indebt later.

To evaluate are approach we will use a real Web Services. Figure (5-1) shows, WSDL file for grade Web Service, this Web Service has two input parameters, Un_Name and Grade.

```
<wsdl:definitions xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
xmlns:tm=http://microsoft.com/wsdl/mime/textMatching/
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime=http://schemas.xmlsoap.org/wsdl/mime/
xmlns:tns="http://tempuri.org/"
xmlns:s=http://www.w3.org/2001/XMLSchema
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http=http://schemas.xmlsoap.org/wsdl/http/
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://tem
puri.org/">
<wsdl:types>

<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.
org/">
```

```
    <s:element name="Un_Name" type="s0: Un_Name " />
      <s:simpleType name=" Un_Name">
        <s:restriction base="xs:string">
          <s:enumeration value=" The University of Jordan " />
          <s:enumeration value=" Yarmouk University " />
          <s:enumeration value=" Jordan University of Science and
Technology
           " />
          <s:enumeration value=" Mutah University" />
          <s:enumeration value=" Hashemite University " />
          <s:enumeration value="Al-Hussein Bin Talal University" />
     </s:restriction>
      </s:simpleType>
    </s:element>
    <s:element name="Un_NameResponse">
      <s:complexType>
        <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="Un_NameResult" type="s:strin
g"/>
        </s:sequence>
      </s:complexType>
    </s:element>
<s:element name="GRADE" type="s0:GRADE" >
<s:simpleType name="GRADE">
            <s:restriction base="s:integer">
               <s:minInclusive value="70"/>
               <s:maxInclusive value="79"/>
            </s:restriction>
          </s:simpleType>
</s:element>
<s:element name="GRADEResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="GRADEResult" type="s:
string"/>
</s:sequence>
</s:complexType>
</s:element>
</s:schema>
</wsdl:types>
<wsdl:message name="Un_NameSoapIn">
<wsdl:part name="parameters" element="tns:Un_Name"/>
</wsdl:message>
<wsdl:message name="Un_NameSoapOut">
<wsdl:part name="parameters" element="tns:Un_NameResponse"/>
</wsdl:message>
<wsdl:message name="GRADESoapIn">
<wsdl:part name="parameters" element="tns:GRADE"/>
</wsdl:message>
<wsdl:message name="GRADESoapOut">
<wsdl:part name="parameters" element="tns:GRADEResponse"/>
</wsdl:message>
<wsdl:portType name="WebService1Soap">
<wsdl:operation name="Un_Name">
<wsdl:input message="tns:Un_NameSoapIn"/>
<wsdl:output message="tns:Un_NameSoapOut"/>
</wsdl:operation>
```

```
<wsdl:operation name="GRADE">
<wsdl:input message="tns:GRADESoapIn"/>
<wsdl:output message="tns:GRADESoapOut"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="WebService1Soap" type="tns:WebService1Soap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="Un_Name">
<soap:operation soapAction="http://tempuri.org/Un_Name"style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GRADE">
<soap:operation soapAction="http://tempuri.org/GRADE" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="WebService1Soap12" type="tns:WebService1Soap">
<soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="Un_Name">
<soap12:operation soapAction="http://tempuri.org/Un_Name" style="document
"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GRADE">
<soap12:operation soapAction="http://tempuri.org/GRADE"style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="WebService1">
<wsdl:port name="WebService1Soap" binding="tns:WebService1Soap">
<soap:address location="http://localhost:62641/WebService1.asmx"/>
</wsdl:port>
<wsdl:port name="WebService1Soap12" binding="tns:WebService1Soap12">
<soap12:address location="http://localhost:62641/WebService1.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Figure (5-1): WSDL for XML schema with restriction

The grade Web Service WSDL file has two input parameters (un_name, grade) first input parameter un_name of type string and has a restriction of type enumeration.

Based on testing technique selection rules introduced in (section 4.3), our approach will chose automatically the syntax testing since the input parameter data type is string and the constraining facet is enumeration, in addition our approach will generate valid and invalid test data based on the following test data generation rules:

If data type = string and constraint facet = enumeration then test data = {value not include in the set, value in the set, null}.

In other hand the second input parameter grade is of type integer and has a constraint facet (minInclusive and maxInclusive) which restrict its acceptable values to be from (70 – 79).

Similarly, since the constraining facets for grade input parameter are minInclusive and maxInclusive our approach will chose boundary value testing technique and generate test data based on it. To generate test data for grade input parameter our approach will follow the following test data generation rule:

If data type = integer and constraint facet = minInclusive and maxInclusive then test data = {min-1, min, min+1, nominal, max-1, max, max+1}.

Table (5-1) shows the generated test data for un_name and grade input parameters using our approach.

Input Parameter	Test Data
Un_name	University Oxford
	Yarmouk University
	Null
Grade	69
	70
	71
	75
	78
	79
	80

Table (5-1): generated test data for un_name and grade input parameters

The generated test data presented in table (5-1) will be sent to the grade Web Service to validate it.

An implementation to examine the applicability of our approach, a tool has been implemented and tested using the previous WSDL file. (Section 5.4) illustrate our tool and discus how it works.

5.4 User Interface

Figure (5-2) presents a snapshot of our tool.

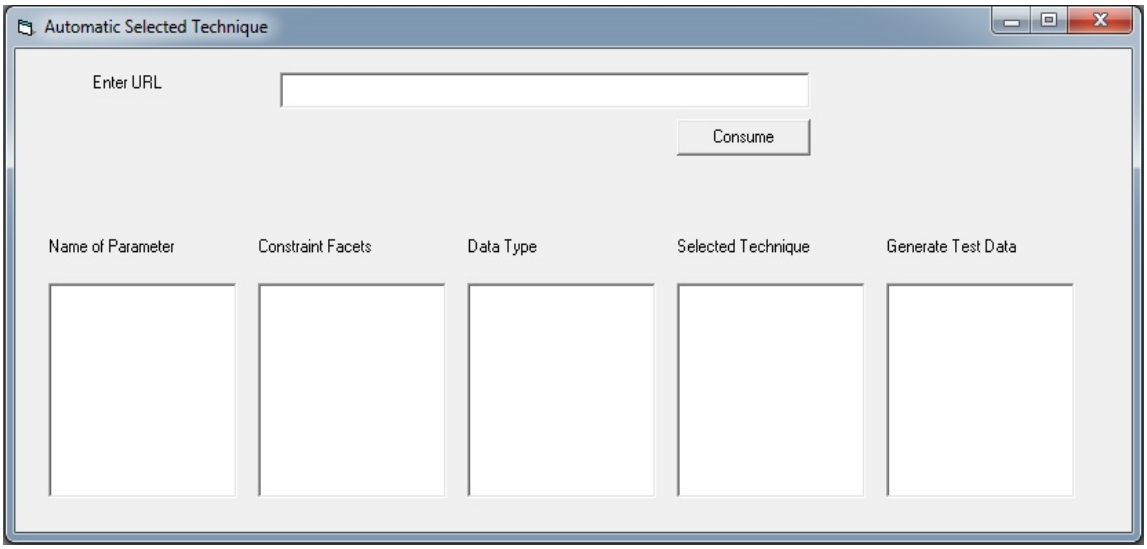

Figure (5-2) represents a snapshot of primary screen

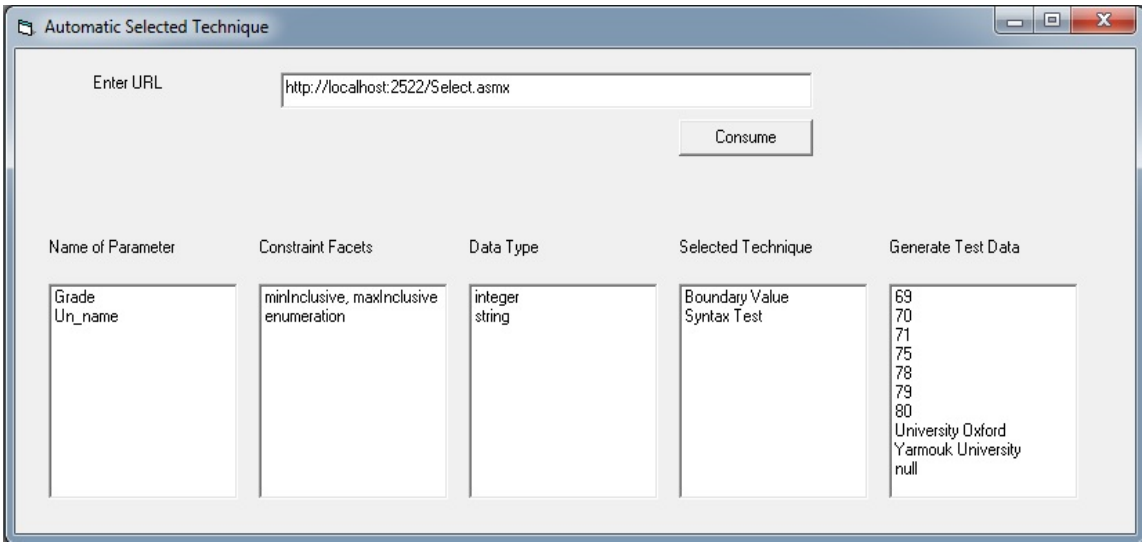

Figure (5-3) snapshot result screen

As shown in figure (5-3), we uploaded the grade Web Service WSDL file and successfully our tool select the suitable testing technique and generate test data accordingly.

5.5 Comparison with similar works

If we want to compare our approach with other similar approaches such as Automated Testing of XML / SOAP Based Web Services (Stepien and Schieferdecker, 2003). We found that (Stepien and Schieferdecker) approach test automatically Web Services, but it needs to determine the appropriate testing technique in advanced in order to test Web services. This takes a long time about 7 to 15 seconds. In the other hand, our approach Automat the selection process using the testing techniques selection rules introduced in section 4.2, then decide which is the most appropriate testing technique to test Web Service, this automated selection process will reduce the time taken in knowing what testing techniques is required for a specific Web services to be about 0.1 second. We applied the two approaches to more than 10 WSDL files and we found the difference in time between the two approaches is very large. Figure (5.4) illustrates Comparing time between the two approaches.

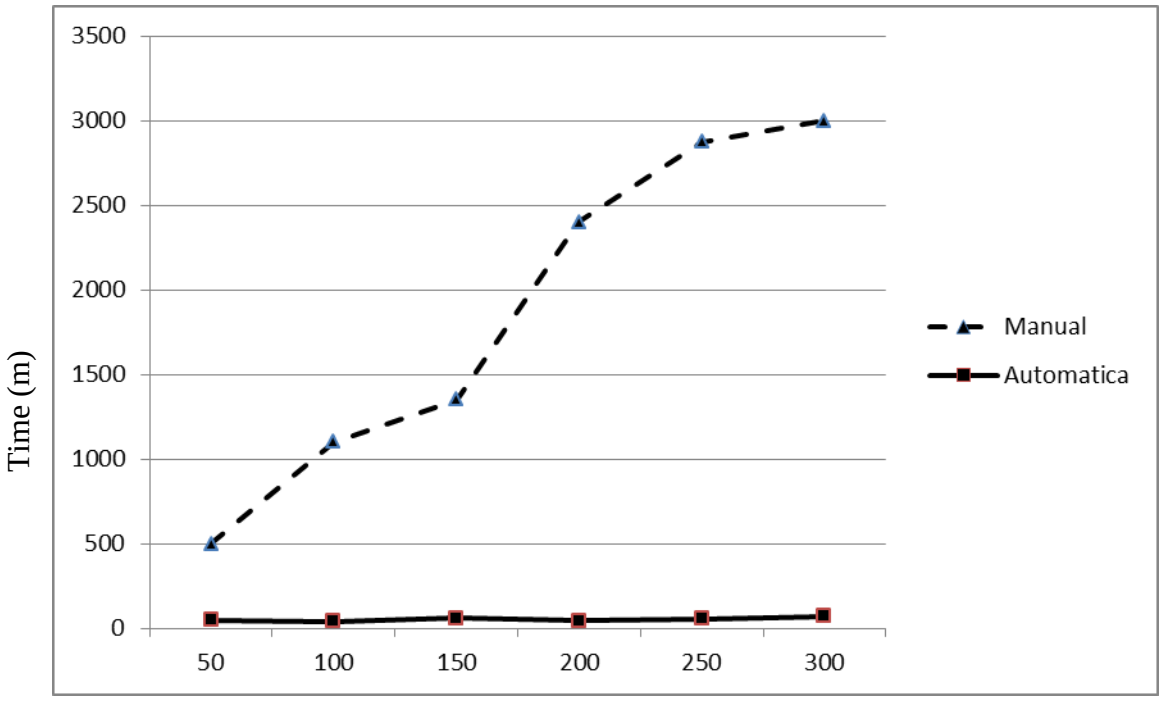

Figure (5-4): Comparing time between the two approaches.

Selection time is the time that is spent after accessing the Web Service to select a suitable technique that is needed to test the parameter which is shown in figure (5-4).

The other approaches need a professionalism to know how the techniques work. While with the automatic selection, any user can use it since the selection ignores the experience and knowledge of the user.

In other hand if we want to compare our approach with other similar approaches such as an approach for WSDL-Based Automated Robustness Testing of Web Services (S. Hanna and M. Munro, 2009).

The authors in (S. Hanna and M. Munro, 2009) focused in the specific attribute for the quality of service, robustness, while in this project no specific attribute is considered. The generated test cases for the robustness are the core of the research but in this project the generated test case used to prove the completeness of the proper selection of automatic selection.

This research proposed the framework to be extendable for any testing techniques and to be generalizing one. The authors in (S. Hanna and M. Munro, 2009) are fixed and deal with only one specific quality of web service.

The authors built schema to specify the rule to generate test data. This project derived own rules from the schema. These rules are generalized to able to select automatically testing technique for data type and constraint facet.

5.6 Summary

This chapter shows the automation of selecting a technique to reduce the time of testing. Our approach tackles the problem that the user needs knowledge about the technique, and it supposes that the normal user does not need to be an expert to test the Web Service. Hence, the selection test technique is automatic and the time of selection is negligible.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The Web Service is a technology that makes the interaction between the applications possible. A lot of testing techniques are provided to test the quality of the Web Service. The WSDL is a file that is generated when the Web Service is published. The XML schema in the WSDL describes the Web Service according to the type of parameters and constraint facets. Testing techniques use XML schema to test the Web Service. The tester takes a long time to select the suitable technique needed to test a specific parameter which has constraint facets that appeared in the XML schema.

The first contribution and the last where achieved by the proposed approach introduce in chapter four, where our approach was able to automatically select the suitable testing technique for a specific parameter. Also this approach saves the wasted time that the service requester spent in choosing the testing technique since our frame work automate the whole process.

Where achieved through the developed selection rules presented in chapter four where our extendible approach enables to accept and delete techniques to test the attribute of the Web Service by simply adding new rules or deleting old ones. In addition, the normal user who does not have a solid knowledge about the nature of testing techniques can use this approach easily.

6.2 Future Works

As a contribution of this research, there are several interesting issues and open problems that require further research and analysis. These may be summarized as follows:

- The tested can be used another time. Using machine learning reduces the time of our previous work.
- Run in back ground, to test simultaneously to give the tester the result when needed.
- The ability to compare between testing techniques and results.

REFERENCES

A. de Melo and P. Silveira, “Improving data perturbation testing techniques for Web services”, Information Science 181 (2011), pp. 600-619.

A. Lyer , (2009) . Evolution and adaptation of web service , A Thesis For the degree of Masters of Science, Queensland University of Technology , A university located in Australian .

B. Stepien & I. Schieferdecker , (2003). Automated Testing of XML SOAP based Web Services, Springer Berlin Heidelberg 01/2003; Source: CiteSeer .

C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini, (2008). Towards Automated WSDL-Based Testing of Web Services , ICSOC '08 Proceedings of the 6th International Conference on Service-Oriented Computing Springer-Verlag Berlin , Heidelberg ©2008.

C. Hart, J. Greenwood, D. Cazzulino, V. G. Aprea, (2005). Beginning visual web programming in Vb. net from novice to professional, Apress; 1 edition ISBN: 1590593596.

C. Mainka, J. Somorovsky, J. Schwenk, (2012). Penetration Testing Tool for Web Services Security, Services (SERVICES), 2012 IEEE Eighth World Congress on , Page(s) 163-170, ISBN: 978-1-4673-3053-4

C. Pik Wah, (2008). Building Reliable Web Services Methodology Composition Modeling and Experiment, A Thesis For the degree of Masters of Science, Chinese University of Hong Kong, A university located in Hong Kong .

D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Chamion, C. Ferris, and D. Orchard, (2004). Web Services Architecture, W3C Working Group Note 11 February 2004, [Retrieved from: http://www.w3.org/TR/ws-arch/].

J. Bloomberg, (2002). Web services testing: Beyond SOAP. [Online], http://searchsoa.techtarget.com/news/846941/Web-services-testing-Beyond-SOAP, August 2002.

J. Garc´ıa-Fanjul, C. de la Riva, and J. Tuya, (2006). Generation of conformance test suites for compositions of web services using model checking, in TAIC PART'06: Proceedings of Testing: Academic & Industrial Conference Practice and Research Techniques, pp. 127–130, Windsor, UK, Aug. 2006, IEEE Computer Society.

K. S. Wagh, R. C. Thool (2013), Web Service Provisioning on Android Mobile Host, International Journal of Computer Applications 81(14):5-11, November 2013. Published by Foundation of Computer Science, New York, USA.

K. Tirghoda, (2012). Web Services Performance Testing Using Open Source Apache Jmeter, International Journal of Scientific & Engineering Research, Volume 3, Issue 5.

M. T. Girish, R. R. Mudholkar, B. T. Jadhav, (2013). ANDROID CLIENT FOR ACCESSING DATABASE WEB SERVICE, proceeding of international conference, IBSS_IETRET 2013.

N. Mansour, H. Fouchal, A. Tarhini, (2005). A Simple Approach for Testing Web Service Based Applications, In proceeding of: Innovative Internet Community Systems, 5th International Workshop, IICS 2005, Paris, France, June 20-22, 2005, Revised Papers Source: DBLP .

P. Ammann & J. Offutt, (2008). Introduction to software testing, Cambridge University Press, Cambridge, UK, ISBN 0-52188-038-1.

S. Hanna, M. Munro, (2009). An approach for WSDL-Based Automated Robustness Testing of Web Services, Springer US, ISBN: 978-0-387-78577-6, pp.1093-1104

Software testing help, [Online], http://www.softwaretestinghelp.com/what-is-boundary-

Value-analysis-and-equivalence-partitioning/. Last visit on 24/09/2013.

S. Salva, I. Rabhi, (2009). Automatic web service robustness testing from WSDL descriptions, Author manuscript, published in "12th European Workshop on Dependable Computing, EWDC 2009, Toulouse : France " .

T. Scholte, D. Balzarotti, E. Kirda, (2012), “Have things changed now? An empirical study on input validation vulnerabilities in web applications”, Computers and Security Journal , ISSN: 0167-4048 and is available at : http://dx.doi.org/10.1016/j.cose.2011.12.013, pp. 344-356.

T. Y. Lee, L. W. Cheung (2010) , "XML Schema Computations: Schema Compatibility Testing and Subschema Extraction", CIKM '10 Proceedings of the 19th ACM international conference on Information and knowledge management Pages 839-848 ,ACM New York, NY, USA ©2010 .

W. T. Tsai, R. Paul, Y. Wang, C. Fan, D. Wang, (2002). Extending WSDL to Facilitate Web Services Testing, High Assurance Systems Engineering,. Proceedings. 7th IEEE International Symposium on , 2002 , Page(s): 171 -172 .

W. T. Tsai, X. Bai, Y. Chen, and X. Zhou, (2005). Web service group testing with windowing mechanisms, in SOSE ’05: Proceedings of the IEEE International Workshop, pp. 221–226, Beijing, China, Oct. 2005, IEEE Computer Society.

W. T. Tsai, X. Wei, Y. Chen, and R. Paul, (2005). A robust testing framework for verifying web Services by completeness and consistency analysis, in SOSE ’05: Proceedings of the IEEE International Workshop, pp. 151–158, Beijing, China, Oct. 2005, IEEE Computer Society.

X. Bai, W. Dong, W. T. Tsai, and Y. Chen, (2005). WSDL-based automatic test case generation for web services testing, in SOSE 2005: Proceedings of the IEEE International Workshop on Service-Oriented System Engineering, pp. 207–212, Beijing, China, Oct. 2005, IEEE Computer Society.

ملخص

الويب سيرفس هي تقنية ظهرت من اجل حل مشكلة إختلاف البيئات البرمجية والمعماريات . وهي ايضا لتسهيل تخاطب البرمجيات فيما بينها حتى لو كانت عن بعد . والويب سيرفس هي عبارة عن مجموعة اوامر غير مرئية للمبرمج موجودة في الجهة المقابلة ويمكن استخدامها من خلال عنوان الويب سيرفس ومعرفة المتغيرات بواسطة نشرها داخل ملف الوسدل. كأي برمجية لابد من فحصها. فقد ظهرت عدة ادوات لفحص هذا النوع من التقنيات . وكانت كل اداة تهتم بنوع خاص فيها من خصائص الويب سيرفس . ولاستخدام اي اداة فحص كان يتوجب على المبرمج فهم طبيعة عمل اداة الفحص وكيفية قراءة ملف الوسدل ثم التأكد من ان اداة الفحص المتوفرة في فعلا هي المناسبة لهذا المتغير في الويب سيرفس . مع وجود العديد من الادوات فإن عملية الاختيار تأخذ وقت من المبرمج. وبالطبع يجب على المبرمج ان يكون ملما بكل الادوات الموجودة.

لذلك جاءت هذه الاطروحة لتوفير الوقت على المبرمج بحيث تختار اداة الفحص بشكل تلقائي بمجرد الوصول الى موقع الويب سيرفس ثم انها لا تقوم بتخيير المبرمج اي اداة سوف يستخدم وانما الخيار سوف يكون من واجب هذه الاطروحة. حتى لو وجد العديد من ادوات الفحص واستجد منها ايضا فإن الاطروحة عندها القدرة على استقبال الجديد وبنفس الوقت تعديل القديم اذا لزم الامر.