Fault-based Web Services Testing
Samer Hanna and Malcolm Munro
Department of Computer Science
Durham University – UK
{samer.hanna, Malcolm.munro}@durham.ac.uk

Abstract

Web services are considered a new paradigm for building software applications that has many advantages over the previous paradigms; however, Web services are still not widely used because Service Requesters do not trust Web Services that were built by others.

Testing can participate in solving this problem because it can be used to assess the quality attributes of Web Services and hence increase the requesters’ trustworthiness. This paper proposes an approach that can be used to test the robustness and other related attributes of Web Services, and that can be easily enhanced to assess other quality attributes.

The framework is based on rules for test case generation that are designed by, firstly, analyzing WSDL document to know what faults could affect the robustness quality attribute of Web Services, and secondly, using the fault-based testing techniques to detect such faults. A proof of concept tool that depends on these rules has been implemented in order to assess the usefulness of the rules in detecting robustness faults in different Web Services platforms.

Keywords: Web Services, WSDL, XML Schema, Fault-based testing, Trustworthiness, Robustness.

1. Introduction

Web Services [6] are considered a new paradigm in building software applications based on the Internet and open standards. This paradigm has changed the way we look at the Internet from being a repository of data into a repository of Services [20]. By using Web Services, companies can ensure that their applications will communicate with those of their business partners and customers even if they are using different programming language or platforms.

A problem that limits the growth of Web Services is the lack of trustworthiness by the Service Requesters because they can only see the WSDL document but not the implementation or source code.

Testing can be used to solve this problem; by assessing the quality attributes of a Web Service under test, the confidence or the trustworthiness of the requesters increase or decrease according to the test results. This will also help requesters to choose between Web Services that are doing the same task.

However, Web Services testing still face many problems like the unavailability of the source code to the Service Requesters and that the traditional testing technique do not cope with the new characteristics introduced by Web Services standards [19]. This paper introduces an approach that can participate in solving these problems. This approach is based on analyzing WSDL documents in order to generate test cases to test the robustness quality attribute of Web Services depending on fault-based testing techniques.

The trustworthiness requirements include security, reliability, safety, survivability, interoperability, availability, and fault tolerance [20], in order to increase Service Requesters trustworthiness of Web Services, all these requirements must be addressed by researchers and practitioners. Robustness is a sub-attribute of reliability [1], so assessing and increasing the robustness of Web Services will contribute to the increasing the reliability and hence the trustworthiness of Web Services.

The robustness quality attribute of Web Services affects other quality attribute such as the fault tolerance to wrong input and security, the test cases design using this paper approach will explain what exactly the quality attribute that is targeted by each test cases.

Also the Web Service platform or the middleware affects the robustness of Web Services because the Web Service platform in the provider side may intercept the SOAP request before reaching the Web Service implementation if that request contains some invalid data, the rule-based test cases proposed by this paper makes a distinction between targeting the robustness of the middleware and the Web Service implementation.

This paper is organized as follows: Section 2 presents a background on relevant concepts and terms used in the rest of the paper. Section 3 discusses the related work in the field of Web Services testing. Section 4 presents the proposed Web Services robustness testing approach. Section 5 discusses the case generation rules in the proposed approach. Section
6 will discuss the experimental results that are used to evaluate the usefulness of the approach. Finally, section 7 will conclude this work and discusses the future work.

2. Background

This section will introduce brief definitions of Web Services and fault-based testing techniques.

2.1 Web Service

There is no universally accepted definition of Web Services, as it has been under debate for quite some time. An extensive literature survey on Web Services showed us that none of current definitions (given by different people and organizations) contained all the relevant characteristics of Web Services. In the context of our work, the definition of Web Service includes those relevant characteristics for our work, as it is defined as follows:

Web Services are network (Internet) based modular applications designed to implement SOA, and support interoperable, loosely coupled, integration of heterogeneous application. Web Services are discovered using UDDI and it has an interface that is describe in WSDL. Other systems interact with the Web Services in a manner prescribed by its description using SOAP, these SOAP messages (as well as all other technologies of Web Services) are based on XML and typically conveyed using HTTP.

2.2 Fault-based Testing Techniques

Fault-based or negative testing is defined as “Testing aimed at showing software does not work” [4].

Testing that a system meets its requirement specifications (validation testing) without applying fault-based testing leave the software system open to vulnerabilities that might not surface until much later in the development cycle or after deployment [5].

In fault-based testing, test cases are written for invalid and unexpected input conditions in order to check how if the system under test will can handle such input gracefully.

Handling the invalid (or manipulated or faulty) input gracefully may include raising an exception with proper error message that describes to the user what happened, while if the system have vulnerabilities to such invalid inputs, then it might reveal so important information that can be used by malicious used to harm the system.

Systems that have an interface which is accessible by public must specially be robust and consequently must have prolific input-validation checks.

The fault-based testing techniques that are important to the research in this thesis are: Interface Propagation Analysis (IPA) which is one of the fault injection techniques [13], Boundary-value based robustness testing [7], equivalence partitioning with invalid partition class [9], and syntax testing [4]. All these techniques belong to black-box testing because it is assumed in this paper that the testing is done by the Web Service Requesters who does not have the source code of the Web Service under test. Table 1 will describe how test data is derived using these fault-based testing techniques.

3. Related Work

Most the researches in Web Services testing focus on specification or validation testing in order to make sure that a Web Service meets a Service Requester’s requirements [12]; however, there are few researches addressed fault-based testing of Web Services.

Since the robustness and other related quality attributes, such as security and fault tolerance, are important to this paper, Table 2 will give a summary of the researches that assess robustness and other related quality attributes using fault-based testing techniques. After surveying the field of Web Services testing, whether fault or validation testing, the following notices have been concluded:

- Some researches do not specify what quality attribute of Web Services they are assessing.
- Different researchers may be assessing the same quality attribute but they describe or call this quality attribute differently.
- Some researches mention that they want increase trustworthiness of Web Services but without specify which specific requirement of trustworthiness they are targeting.
- Some researches specify that they do negative testing but they do not specify how the negative or faulty test data was generated, in other words which testing techniques have been applied.

Table 1. Test data generation method in different fault-based testing techniques

<table>
<thead>
<tr>
<th>Testing Technique</th>
<th>Test Data Derivation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPA</td>
<td>Randomly perturbing</td>
</tr>
</tbody>
</table>
This section will describe an approach for assessing the robustness, input manipulation vulnerability, and fault tolerance to wrong input quality attributes of Web Services. Figure 1 describes the overall architecture of the Web Services robustness testing approach proposed in this paper. The components of the architecture in Figure 1 are described as follows:

- **Web Service platform** is the platform or middleware that the Web Service Provider is using for his Web Service implementation. Examples of Web Service platforms are Axis [3] and GLUE [15].
- **Test case generation rules** are the rules that is proposed by this paper for test case generation, these rules depends on the faults that affects the robustness quality attribute of Web Services and the fault-based testing techniques. Test case generation rules will be discussed in section 5.
- **Web Services test case generator** is the component that is responsible for generating test cases based on the WSDL document of the Web Service under test and the test case generation rules.
- **Test results** component is an XML document that describes the test data together with the actual response of the Web Service under test for each of the test data in each test case.
- **Web Service implementation** is the source code of the Web Service that is written by the Web Service Provider.
- **Automatic Client Generator** is the component that is responsible of building a client to the Web service under test and invoking the Web service under test using the test data provided by the Web Service test case generator component.

After defining the different components of the overall architecture in Figure 1, the interaction between those components will be described:

1. Test case generation rules are designed based on the faults that may affect the robustness of Web Services depending on the specification inside WSDL.
2. Web Service Provider deploys his/her Web Service implementation in a Web Service platform.
3. The Web Service test case generator component uses the test cases generation rules in step 1 and the WSDL document of the Web Service in step 2 to generate the Web Service test case.

### Table 2. Researches on Fault-based Testing of Web Services

<table>
<thead>
<tr>
<th>Research</th>
<th>Testing Techniques</th>
<th>Quality Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offutt &amp; Xu, [10]</td>
<td>Boundary value testing, data perturbation, mutation testing, SQL injection.</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Xu, et al. [16]</td>
<td>XML Schema perturbation</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Zhang &amp; Zhang [20]</td>
<td>Boundary value testing together with faulty data perturbed from boundary value, and using WSDL for test case generation</td>
<td>Reliability (correctness and fault tolerance to faulty input data)</td>
</tr>
<tr>
<td>Siblini &amp; Mansour [11]</td>
<td>WSDL-based testing and Mutation testing</td>
<td>Unspecified</td>
</tr>
<tr>
<td>Looker, et al. [8]</td>
<td>Fault injection with white box manner</td>
<td>Dependability</td>
</tr>
</tbody>
</table>

4. **Web Services Robustness Testing Approach**
4. The automatic client generator will generate a client to invoke the Web Service deployed in step 2 using the test case developed in step 3 and then generate the test results document accordingly.

5. Test Case Generation Rules

Test case generation depends on the input parameters datatype XML Schema based specification inside WSDL and the robustness related faults that may be resulted based on violating this specification.

Appendix 1 contains a schema for the test case generation rules proposed by this paper (See Table 3).

A brief description of the attributes or components of the schema in Table 3 is as follows:
1. The ID attribute is a unique identifier for different rules.
2. WSDL component test data is based on, since the test cases depending on the information inside WSDL, this attribute specifies the WSDL component that the current test case is based on.
3. The fault attribute is the fault that is based on a certain element inside WSDL and that the current test case assumes to detect.
4. Traditional testing technique describes the fault-based testing technique (See section 2.2) that is used to generate test data to assess the fault in step 2.
5. Traditional test data generation rules describes how the test data is generated depending on the testing technique in step 3.
6. Valid/Invalid attribute used to specify if the test data are valid or invalid test data.
7. Web Service datatype, this attribute describes the XML Schema-based datatype of the input parameter of the Web Service operation under test. XML Schema datatypes are discussed in [14]. According to [14] XML Schema datatypes can be categorized to: primitive or derived from primitive simple datatypes, user-derived datatypes, and complex datatypes.
8. Web Service test datatype, this attribute defines the datatype of the test data used in this test case. The datatype of the test data in not always the same as the Web Service datatype because some test cases use for example integer input for an operation that accepts a string as input in order to test if the operation with produce a proper or graceful exception or not.
9. Web Service test data, this is the actual data that is used to in the current test case.
10. Expected output, this attribute specifies the robust expected SOAP response or SOAP fault of the Web Service under test based on the current test case.
11. Quality attribute assessed, this attribute specifies the quality attribute targeted by the current test case. This research mainly concerned with the robustness quality attribute, however, other quality attribute, such as security, may also be tested by the same test case.

6. Implementation and Results

A proof of concept tool has been implemented using Axis 1.4, Tomcat 6.0 [2], and Java 1.5.0_06. This tool can do the following:
- Generate test data for a certain Web service depending on its WSDL only and by using the rules described in the previous Section.
- Store the resulted test data in an XML file so that it can be used by any consumer or provider regardless of the platform or the middleware he is using. Figure 3 shows only part of the test data that was generated using our tool for an input parameter of type integer.
- Automatically build a consumer or client to the Web service under test using the test data XML file generated in the previous step.
- Generate test cases by using the test data in the previous test data XML file to invoke the Web service under test using the client generated in the previous step and then store the response of the Web services.

Figure 1. Overall architecture of the Web Services Testing Approach
Web service under test together with the test data used in a new XML file.

This tool then was used to generate test cases to assess the robustness of a Web Service based application called book finder; this application used to search for the best price book in different suppliers. The tool was successful in generating test cases based on the WSDL documents of this application and the test cases succeeded to assess if a certain operation can handle violating its input parameter XML Schema specifications.

Also we have tried to assess the affect of the Web Service platform on the robustness of a certain Web Service, to accomplish this goal; the same Web Services were deployed in Axis and then in GLUE Web Service platform and then the SOAP response to different test cases has been analyzed to assess which of the platform is robust.

An example of the results obtained by these experiments; we deployed a simple Web Service that accepts a simple double and returns also a simple double datatype in Axis platform then in GLUE: table 4 give a summary of the responses generated by the Web Service in different platforms.

The NullReplacement test case shows that both Axis and GlUE are not robust when the input parameter datatype is replaced by null. Axis in not robust because the SOAP fault indicate that the operation does not exist while this operation exists in the WSDL document of the Web Service under test. Glue was also not robust to null input because the fault string that is meant to give the Service Requester the cause of the problem or fault was empty. GLUE was also not secure to the null input because give the stack trace to the Web Service Requester in the SOAP fault.

As for the StringReplacement test case, Axis was robust because it sent a SOAP fault that contains a proper fault string that described that the Web Service is suppose to accept a double rather than a string. GLUE was not robust on this test case because it sent a SOAP fault with fault string “For input string <test>” where “test” is the random string given to the Web Service, it is clear that this fault string does not explain to the Service Requester what is the fault that happened. GLUE was also not secure for the same reason of the NullReplacement test case.

7. Conclusions and Future Work

Web Services still not widely used because Service Requesters do not trust Web Services that were built by others. To solve this problem all the trustworthiness requirements such as reliability, safety, security, interoperability, etc. must be addressed by researchers and practitioners.

This research proposes an approach to assess the robustness quality attribute of Web Services. Robustness is a sub-attribute or requirement for the reliability attribute, so improving the robustness will improve the reliability and hence the trustworthiness of Web Services.

Test cases are destined in this paper approach based on the XML Schema based input parameter specification inside WSDL and the robustness faults that may affect a Web Service based on violating these specifications.

Assessing the robustness quality attribute contributes to the assessment of other quality attributes such as security and fault tolerance to wrong input.

A tool has been implemented that can help the Service Requester to assess the robustness of a Web service based only on its WSDL.

The robustness of a Web Service may be affected by the Web Service platform or the middleware that this Web Service is deployed in. The test cases designed in this paper distinguish between testing the robustness of the platform and testing the robustness of the Web Service implementation.

Future work will assess other Web Service trustworthiness requirements especially more research on the security of Web Services.

Appendix 1 – Rules for Test Case Generation

This appendix contains: schema for test case generation rules (Table 3)

Table 3. Schema for Test Case Generation Rules
<table>
<thead>
<tr>
<th>Attribute</th>
<th>type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>String</td>
<td>Identifier or reference of the rule</td>
</tr>
<tr>
<td>WSDL Component(s) test data is based on</td>
<td>enum</td>
<td>The WSDL component(s) this test data is based on which could be the input parameter datatype or the constraining facets for the input parameter datatype</td>
</tr>
<tr>
<td>Fault</td>
<td>enum</td>
<td>The fault that the test data suppose to detect</td>
</tr>
</tbody>
</table>
| Traditional Testing Technique                 | enum      | The traditional testing technique used in the rule, Testing Technique := EP | RT | IPA | ST  
|                                              |           | Where EP = Equivalent Partitioning                                          |
|                                              |           | RT = Robustness Testing                                                     |
|                                              |           | IPA = Interface Propagation Analysis                                         |
|                                              |           | ST = Syntax Testing                                                         |
| Traditional test data generation rule         | String    | Description of how the test data is generated using the traditional testing technique choose |
| Valid/Invalid                                 | enum      | whether the test data chosen valid or not                                   |
| WS Datatype                                   | Datatype  | Defines the Web service datatype of the input parameter tested. Where WS_datatype ::= String | Numeric | Time-Date | null |
| WS Test Datatype                              | Datatype  | Defines the Web service datatype of the test data which might be the same as the Web service datatype or different. |
| WS test data                                  | depends on WS Test Datatype | Defines the actual data used for testing                                      |
| Expected output                               | String    | Defines what is the expected response SOAP message of the Web service under test |
| Quality attribute(s) assessed                 | enum      | Defines the quality attribute this test data aims to assess               |

8. References