Platform Effect on Web Services Robustness Testing

Samer Hanna and Amer AbuAli
Faculty of Information Technology, Philadelphia University, Jordan

Abstract: Web services are considered a new paradigm for building software applications that has many advantages over the previous paradigms. Web service testing is a very important factor to increase or decrease the requester's trust of a given Web service. However, Web service testing still facing many problems; one of these problems is that the platform where a Web service is deployed may affect the testing results; this research proves this claim by demonstrating how the robustness and security quality attribute of a Web service may be affected by the platform this Web service is deployed on. This will be accomplished by comparing the responses of Web services accomplishing the same task deployed on different platforms, namely, axis, GLUE and Microsoft .NET. This study approach proves that Web services robustness and security testing do actually depend on the platform.

Key words: Web service, platform, testing, quality attributes, robustness

INTRODUCTION

Web Services are a new paradigm for building distributed software applications. They have many advantages over previous paradigms such as increasing the interoperability between heterogeneous applications and facilitate sharing data and information between an enterprise, its branches and customers, even if they are using a different platform, programming language or operating system.

However, the Web Services paradigm is still not widely adopted by companies and individuals because of many problems; very few of these problems are:

• Trustworthiness challenge, service requesters do not trust Web services built by others given the fact that they can not see the implementation of a given Web service and they can see only the description WSDL (Web Services Description Language) of a Web service before integrating it with their application
• The difficulty choosing between services accomplishing the same requester requirement given that WSDL does not include Quality of Service (QoS) specification
• The difficulty of locating appropriate and reliable service that satisfies requester's application requirement

Testing is one aspect of solving these problems because it can be used to assess the quality.

Attributes of Web Services and hence increase or decrease the trust based on the test results.

Web service can not be accessed directly by service requesters but they are accessed by software applications that are written by them. These software applications are called.

Web services-based application (or client applications) that rely on services available on the Internet.

The components that participate in a typical Web service invocation in a Web service-based application are: service provider, Web service implementation, WSDL, server stub (or skeleton), service requester, client application and client stub (or proxy) and Web service container. The interaction among these components to send and receive Simple Object Access Protocol (SOAP) messages will be discussed.

There are many tools or platforms that can create a client stub and a server stub based on WSDL and manage the creating and sending of SOAP messages over the Internet. These tools are called SOAP-based Web services platforms or SOAP engines. Examples of these platforms are Apache Axis (Apache Software Foundation). Web Services – Axis, http://ws.apache.org/axis), GLUE (Web Methods, http://www.webmethods.com/Developers) and .NET (http://msdn.microsoft.com/en-us/netframework/default.aspx).

There exists some research and tools in Web services testing that tried to solve the Web services problems, however, non of the research or tools in this domain, as far as the author's knowledge, addresses the problem that the Web service testing may be affected by the platform where a Web service is deployed.
This study focuses on this aspect of Web services testing by comparing the robustness and security of the same Web service deployed in three different platforms, namely, Axis, GLUE and .NET.

The main contribution of this study is providing an analysis of the effect of the Web Service platform on the robustness and security quality attributes of Web services by analyzing the responses of Web services deployed on different platforms for the same SOAP request.

BACKGROUND

Since Web services testing in this research is based on analyzing WSDL documents and then using the operation’s specification inside WSDL to invoke the Web service under test, next section will discuss WSDL documents and the Web services invocation mechanism, respectively (AbuAli et al., 2010).

WSDL: WSDL is an XML based description of a Web Service and it plays a key role because:

- The provider publishes a description (WSDL) of his Web Service to the broker
- The broker advertises descriptions of Web Services
- The application builder uses the information inside the WSDL description to bind to the provider’s Web service

While, SOAP gives the application builder just the format of the envelope of the messages that are used to exchange information with the provider, WSDL gives the application builder meta data that includes:

- A description of all the operation a Web Service provides
- The syntax of the input and output messages to invoke those operation (data format)
- What messaging or transport protocol should be used to access the needed Web Service
- Where to send the messages (Web Service’s location)
- Shortly, WSDL gives the application builder a formal description of all the information needed to invoke a Web service

WSDL describes a Web Services by using the following major elements: portType, message, parts, types, binding, port and service. The important elements for this study are message, types and also the message’s sub-element part.

The message element defines the format of a message which includes the message name and also a collection of part elements. There are three kinds of messages; input, output or fault message. In document-centric Web service messages are modelled as XML document, while in RPC centric Web services, messages are modelled as parameters for a method (structured programming concepts). This study is concerned only in input messages that are used as parameters in RPC centric Web services.

The part element defines the part name and its type and it is corresponded to the method parameters in structured programming.

The type’s element defines all the datatypes that are referenced in the message’s part element.

Web services invocation: The process of invoking a certain operation provided by a Web services implementation includes the following:

- The client application calls the client stub using application specific datatype (depending on the programming language this application is written in)
- The client stub will convert this local invocation into a SOAP request (marshalling)
- The SOAP request is sent over the Internet (using HTTP) to the required Web service container
- The Web service container (server) receives this SOAP request and then hands it to the skeleton (server stub)
- The skeleton converts the SOAP request into application specific data and send it to the Web Service implementation (depending on the programming language the Web Service implementation is written by)
- The Web service implementation performs the requested operation that it was asked to perform by the skeleton
- The result of this operation will be handed to the skeleton
- The skeleton convert this application specific result into a SOAP response (or SOAP fault if the Web Service implementation raised an exception or anything else went wrong)
- The SOAP response (or SOAP fault) message is sent to the client stub using the Internet (over HTTP)
- The client stub converts the information inside the SOAP response (or SOAP fault) message into an application specific information (that can be understood by the client application) and sends it to the client application
- Axis, GLUE and .NET are examples of platforms that are responsible for generating and transferring SOAP messages over the internet.
XML schema datatypes: XML Schema datatypes can be categorized to: built-in primitive datatypes, derived from built-in primitive datatypes, user-derived datatypes, list datatypes, union datatypes and complex datatypes.

This study is concerned mainly with primitive and derived from primitive datatypes and they are categorized to numeric, string, date-time and boolean (Table 1).

Web services testing: Web services testing in this study refers to assessing the ability of a Web Service to handle invalid input by the service requester or robustness testing. Testing Web services for robustness is a difficult task and existing development tools of Web services do not provide any mean to assess Web services robustness in the presence of invalid inputs (Laranheiro et al., 2008).

Increasing Web services robustness is one factor of increasing Web services trustworthiness, other factors may include availability, security, fault tolerance and other quality attributes related to web service (Hann & Alawneh, 2009) for a complete list of the quality attributes that affect Web services). The trustworthiness of Web-Service software is considered the paramount factor that will decide the success of the Web Services paradigm (Zhang and Zhang, 2005).

Since, testing is performed to support quality assurance then it is normal to use it with Web Services in order to increase their quality and hence increase the service requester’s and the service provider’s trust.

Using testing to assess the quality attributes Web Services has many advantages such as:

- Help the service requesters to choose between Web services that accomplish the same task depending on the quality attributes that concerns each service requester.

However, Web services testing also facing many problems and difficulties such as:

- There is lack of technologies for Web services verification (Zhang and Zhang, 2005)
- Current methods and technologies cannot guarantee Web service trustworthiness (Zhang, 2005; Tsai et al., 2005)
- Testing may be affected by the platform or middleware where a Web service is deployed on

This study main target is to demonstrate the middleware dependence difficulty.

**RELATED RESEARCH ON WEB SERVICES ROBUSTNESS TESTING**

There are many researches concerning Web services robustness testing, some of them are

Laranheiro et al. (2009) proposed a mechanism and a tool that can automatically fixes and improves the robustness of Web services implementations.

Laranheiro and Vieira (2009) also demonstrated how robustness testing can be integrated in the test driven development process to improve Web services robustness.

Noikajana and Suwannasart (2009) used pair-wise testing technique to generate test cases for Web services robustness testing using a Web Service Semantics Language (WSDL-S) and the Object Constraint Language (OCL).

Vieira et al. (2007) proposed an approach for evaluating the robustness of Web services based on a set of robustness tests (invalid calling parameters for the Web service under test).

Martin et al. (2006) presented a framework for automatically generating and executing Web services requests to assess the robustness quality attribute.

Jiang et al. (2009) used data perturbation which is considered a black box testing technique to generate test cases for Web service robustness.

AbuAli et al. (2010) tested the vulnerabilities of websites with similar approaches used to test web services.

After a comprehensive literature review in the Web service testing domain, it has been found that there is no research considered the effect of changing the Web
service platform on the tensing process of Web service, in general and specially testing the robustness quality attribute of Web services.

DEMONSTRATING PLATFORM'S EFFECT

The implementation of this section's experimental examples uses Java version 1.5.0.06, Axis 1.4, GLUE 1.2, Tomcat 6 server, Microsoft .NET framework 2008 and HTTP server.

To assess the effect of the Web service platform on Web services robustness testing, the equivalent partitioning testing was used for the datatype partitions in Table 1. For the numeric datatypes class, the double datatype was chosen to represent the datatypes in this class. In a similar way for the string datatypes, the string datatype was chosen to represent the datatypes in this class, for the date-time datatypes, the date datatype was chosen to represent datatypes in this class and finally for the Boolean datatypes, the boolean datatype, the only element in this class was used.

The method of generating test cases for Web services based on the Web services's operation input datatype is fully descibed in our previous research (Hanna and Munro, 2009).

Four different simple Web Services that accepts double, string, date and boolean respectively were deployed in the Axis platform and then in the GLUE and .NET platforms to compare the responses of Axis, GLUE and .NET for the same test cases.

Table 2-5 show the comparative responses using Axis, GLUE and .NET platforms.

Table 2 uses the following abbreviations:

FMP: Fault Message with Proper fault string sent by the Web Service platform for changing the datatype of the input parameter
RM: Response message
FM: Fault message

The rule for the test cases generation to assess the robustness quality attribute of Web services (String Replacement, Date-Time Replacement, Boolean Replacement and Null Replacement) (same 2) is the replacement of the double datatype that is expected by the web service by string, date, boolean and null, respectively.

The same discussion applied for Table 2 can be made for Table 3-5.

RESULTS

The results obtained for the four examples in Table 2-5 will be discussed separately as follows:

The Web service that accepts primitive double input datatype (Table 2):

- Axis and .NET are robust when applying the String_Replacement test case while GLUE is not. The reason is that both Axis and .NET return a fault message with a proper fault string while GLUE returned a fault message with a fault string that does not describe the fault that happened, the fault string is For input string <test> where test is the string that is used to replace the actual double datatype

- Axis and .NET are robust when applying the Date-Time_Replacement test case while GLUE is not. The reason is that both Axis and .NET return a fault message with a proper fault string while GLUE returned a fault message with a fault string that does not describe the fault that happened, the fault string is For input string <2010-02-20T00:00:00.000Z> where 2010-02-20 is the date that is used to replace the actual double datatype

- Axis and .NET are robust when applying the Boolean_Replacement test case while GLUE is not.

<table>
<thead>
<tr>
<th>Test case</th>
<th>Axis 1.4 response or fault</th>
<th>GLUE 1.2 response or fault</th>
<th>.NET 2008 resp. or fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringReplace</td>
<td>FMP</td>
<td>FM with fault string: For input string &lt;test&gt; detail element containing stack trace</td>
<td>FMP with stack trace</td>
</tr>
<tr>
<td>Date-TimeReplace</td>
<td>FMP</td>
<td>FM with fault string: For input string &lt;2010-02-20T00:00:00.000Z&gt; detail element containing stack trace</td>
<td>FMP with stack trace</td>
</tr>
<tr>
<td>BooleanReplace</td>
<td>FMP</td>
<td>FM with fault string: For input string &lt;true&gt; detail element containing stack trace</td>
<td>FMP with stack trace</td>
</tr>
<tr>
<td>NullReplace</td>
<td>FMP</td>
<td>FM with empty fault string detail element containing stack trace</td>
<td>FMP with stack trace</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test case</th>
<th>Axis 1.4 response or fault</th>
<th>GLUE 1.2 response or fault</th>
<th>.NET 2008 resp. of fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumericReplace</td>
<td>FMP</td>
<td>RM with a null replacing the Numeric value passed</td>
<td>FMP with stack trace</td>
</tr>
<tr>
<td>StringReplace</td>
<td>FMP</td>
<td>RM with a null replacing the String value passed</td>
<td>FMP with stack trace</td>
</tr>
<tr>
<td>BooleanReplace</td>
<td>FMP</td>
<td>RM with a null replacing the Boolean value passed</td>
<td>FMP with stack trace</td>
</tr>
<tr>
<td>NullReplace</td>
<td>RM (null accepted)</td>
<td>RM (null accepted)</td>
<td>FMP with stack trace</td>
</tr>
</tbody>
</table>
The reason is that both axis and .NET return a fault message with a proper fault string while GLUE returned a fault message with a fault string that does not describe the fault that happened, the fault string is for input string <true> where true is the Boolean that is used to replace the actual double datatype.

- Both Axis and GLUE are not robust when applying the Null_Replacement test cases generation rule that replace the input parameter with null. Axis is not robust because it returned a fault message with a fault string that does not describe the fault that happened, while GLUE is not robust because it returned an empty fault message which means that the Service Requester will not know what that fault that has happened.

- GLUE was not secure when applying the String_Replacement, Date-Time_Replacement, Boolean_Replacement and Null_Replacement. The reason is the GLUE returned a SOAP fault message with a stack trace inside the detail element of this message.

- Axis was secure in all the test cases applied because no stack trace was generated in any of the SOAP fault messages, where stack trace information can be used to attack a given Web service by malicious requesters.

- .NET was not secure in all the test cases applied because a stack trace was generated in all the SOAP fault messages for each test case applied.

### The Web service that accepts primitive date input datatype (Table 4):

- Axis and .NET are robust when applying the Numeric Replacement, String Replacement and Boolean Replacement test case while GLUE is not. The reason is that GLUE returned a SOAP response message where the numeric, string and boolean inputs are converted to null.

- Axis and GLUE are not robust when applying the Null Replacement while .NET is robust, the reason is that both Axis and GLUE accepted the null input and did not return a SOAP fault while .NET returned a fault message with proper fault string.

- .NET, however, is not secured for all the test cases because, it returned a stack trace.

### The Web service that accepts primitive boolean input datatype (Table 5):

- Axis, GLUE and .NET are robust when applying the Numeric Replacement, String Replacement and Date-Time Replacement test case because both Platforms behave as expected by these test cases by sending a SOAP fault with proper fault string that describe the fault happened of changing the input datatype.

- Axis and GLUE are not robust when applying the Null Replacement because both Axis and GLUE accepted the null input and did not return a SOAP fault while .NET returned a fault message with proper fault string.
Table 6: Comparison of robustness and security between Axis and GLUE

<table>
<thead>
<tr>
<th>Platform</th>
<th>TC</th>
<th>RF</th>
<th>SF</th>
<th>Fault description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td>Handling null input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Handling changing the input datatype</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Handling null input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Stack trace in the SOAP fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Null accepted as input</td>
</tr>
<tr>
<td>GLUE</td>
<td>16</td>
<td>13</td>
<td>4</td>
<td>Handling null input</td>
</tr>
<tr>
<td>.NET</td>
<td>16</td>
<td>1</td>
<td>15</td>
<td>Stack trace in the SOAP fault</td>
</tr>
</tbody>
</table>

- .NET, however, is not secured for all the test cases because it returned a stack trace.

Table 6 summarizes the results obtained by showing the number of the Test Cases, the number of Robustness Failures, Security Failures and the description of the fault the led to these failures for the Axis, GLUE and .NET Web services platforms.

Where, TC is number of test cases, RF is robustness Failure, SF is security failure.

The conclusions are:

- .NET is the most robust platform with only 1 robustness failure
- GLUE is the worst platform in terms of robustness quality attribute because it had 13 robustness failures
- Axis is the most secure platform with no security failures
- .NET is the worst platform in terms of security quality attribute because it had 15 security failures

CONCLUSIONS AND FUTURE WORK

Web services testing is very important process to increase the trustworthiness and adoption of Web service, however, this domain still facing many problems such as the effect of the platform where a Web service is deployed on testing process. This study demonstrated this effect by analyzing the SOAP responses or SOAP faults generated by the same Web services deployed in different platforms, namely, Axis, GLUE (java based platforms) and Microsoft .NET.

The conducted experiments have demonstrated that the Web services robustness and security testing are indeed affected by the platform.

In terms of robustness, .NET was the most robust platform with only 1 robustness failure. And in terms of security, axis is the most secure platform with no security failures.

Future work will demonstrate whether there are other Web services quality attributes may be affected by the platform where a Web service is deployed.

REFERENCES


