Test Case Selection for Regression Testing of applications using Web Services based on WSDL Specification changes

Prerna Singal
CSE/IT Dept,
ITM University, Gurgaon
prernasingal@yahoo.com

Anil K Mishra
CSE/IT Dept,
ITM University, Gurgaon
anilkrmishra@itmindia.edu

Latika Singh
CSE/IT Dept,
ITM University, Gurgaon
latikasingh@itmindia.edu

***Abstract*— there is much enthusiasm around web services in today's world. Web Services take the advantage of internet to communicate between two electronic devices connected via a network. Testing a Web Service is a challenge as the Service Requester does not have the source code and somehow needs to fully test the impact of changes on his application. Regression testing verifies the integrity of the application and makes sure that the changes have not introduced new software errors. Our approach involves the parsing of the WSDL XML file to extract information regarding the operation name, input message and output message. Both the original and changed XML files for the web service are parsed to extract their respective information from the port type and message element of WSDL. Then, we generate a hash table form the extracted information for both the original and delta WSDL. We pass the hash tables to a Comparator as input, which then compares the hash tables and generates the operation changes as output. In the last step test cases are selected for regressing testing of the changed web service based upon the changes in operations provided by the comparator.**

Keywords—Web Services; Regression Testing; Hash Table; WSDL.

I. INTRODUCTION

There is much enthusiasm around web services in today's world. Web Services take the advantage of internet to communicate between two electronic devices connected via a network. Web Service is essentially a software system whose function is to support interoperable machine-to-machine interaction for transmitting data in a network [1]. They are standardized web applications which publish their specification in the UDDI directory that interact with other web applications for the purpose of exchanging data. Web services use five main standards to communicate over the network: Web Services Description Language (WSDL) [3], Extensible Mark up Language (XML), Hyper Text Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP) [2] and Universal Description, Discovery, and Integration (UDDI).

Web Service has a Service Requester and a Service Provider. The application software which requests data is called a Service Requester, and the application software that would process the Requester's request and provide the data is called the Service Provider. The source code for the web service is with the Service Provider. Service Requester has only the WSDL. So, whenever a change occurs in the web service, retesting of web service is required. Testing is challenge for the Service Requester as he does not have the source code and needs to fully test the impact of changes on his application. Web services do not have a Graphical User Interface. Instead they use a programmable interface for exchange of messages [7].

Web Services evolve over time as any other software application. Likewise, we need to perform Regression testing whenever there is change in the Web Service. Regression testing is done to ensure that the changed version of the Web Service is working as desired and it is still performing all the operations correctly. Regression testing verifies the integrity of the application and makes sure that the changes have not introduced new software errors [6]. Regression testing of Web Services poses a greater challenge of providing maximum test coverage to the integration with the application ensuring minimum number of test cases with minimal cost of appraisal and risk.

[4] Ha and Park have Proposed an ontological technique where they apply user level Quality of Service (QoS) that provides two different levels to serve Web service with proper quality by contribution value. Till date, great effort has been put into the research of testing Web Services [7] for integration with other applications. Regression Testing of Web Services is one such area which needs to be researched and improve the efficiency and time of testing. Many of the existing approaches for Regression testing of Web Services are based on Code based testing. They have their limitations as the Service Requester does not have the source code for the Web Service to be tested. A specification based approach of testing of web services has been proposed by Masood and Nadeem [8]. We have extended their approach and suggested some changes in the way XML are stored in order to ease the regression test case selection of Web Services.

Specification based approach utilizes the WSDL for a Web Service for selecting the test cases to be executed to retest the web service. WSDL is an XML-based interface definition language. It describes the functionality and operations offered by a web service. WSDL contains all the details regarding the operations that can be performed by the web service –

ISBN:978-1-4799-8890-7/15/$31.00 ©2015 IEEE

Operation Name, Port Type, Message Types, Input Message, Output Message and Binding. Port Type contains the information regarding the operation names available in the web service and the messages associated with it. Messages have the data elements of the operation like parameter names and their types and the output type of the operation.

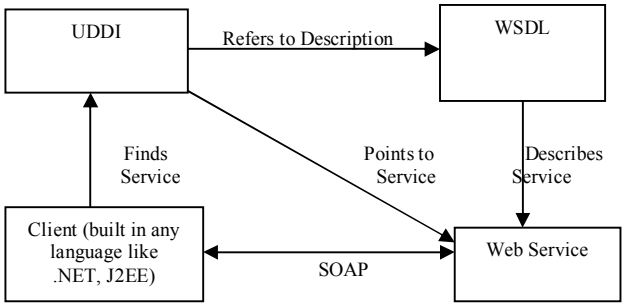

Fig. 1: Web Service Implementation

Our approach involves the parsing of the WSDL XML file to extract information regarding the operation name, input message and output message. Both the original and changed XML files for the web service are parsed to extract their respective information from the port type and message element of WSDL. Then, we generate a hash table form the extracted information for both the original and delta WSDL. We pass the hash tables to a Comparator as input, which then compares the hash tables and generates the operation changes as output. In the last step test cases are selected for regressing testing of the changed web service based upon the changes in operations provided by the comparator.

Test Cases are associated with the operations in the web services directly. Our test suite maintains the test cases along with the operation names. The Regression Test selector chooses only those test cases where there is a change in the operation name or related values or added a new operation. Other test cases for which there is no change in the system may also be selected for regression testing. This is done to ensure that the application is behaving as is before the changes. All parts of the system are covered in the Regression Testing which ensures safety [10]. The Safe Regression Test Suite covers all the test cases for the system to be tested properly. It covers the test cases for the parts which have changed as well as the parts that have not changed in the system.

Rest of the paper has been written to describe the proposed approach in detail. It is organized as follows: Section II is dedicated to literature and Section III outlines the details of the proposed approach. Section IV provides a conclusion for the paper and Section V includes any further directions for research and discussion.

II. Related Work

Web services evolve with the change in technology and industry practices. As the changes occur, Regression testing becomes an important and very expensive activity in order to ensure that these changes will not disrupt the existing functionalities of the Web application software system. Regression testing can be:

1. **Code Based:** Code based techniques consider the source code changes made in the application software. Test cases are selected based on changes made in the code for the application. Thus, code based techniques become very specific to the programming language used to develop the source code.
2. **Model Based:** These techniques generate regression test cases by looking into different system models. Most of the model based techniques are based on Unified Modelling Language (UML) models.
3. **Specification Based:** These techniques are based on the specification and changes in features provided by the customer. Regression Testing is done to ensure that all the specifications as laid by the customer are met and the system is safe.

Ruth et al. Based their research on web services made in JAVA. They proposed a safe regression testing technique for such services. Application under test was analysed for static and dynamic analysis of code. A control flow Graph (CFG) based on the JAVA Code and named it Java Interclass Graph (JIG). Then, JIG is created for both the original and the new code. A comparison is made between the two JIGS' and potential dangerous edges are identified. Finally, test cases are selected based on the dangerous edges selected. They have validated the approach with a simulation tool used to identify the dangerous edges [12].

Ruth et al. extended their work on JAVA specific web services to all the web services in general. They used a similar framework for safe regression testing for generic web services. A Control Flow Graph (CFG) was created for all the web services under test. Then, a comparison is done between the CFG for the old and new web service to highlight the changes in the Web Service. They also proposed to publish test cases [13] along with the WSDL specification, which they find could be helpful in selection of test cases for regression testing.

Penta, et al. proposed a toolkit that generated XML encoded test cases for Regression Testing [7]. Test Cases were used as a Contract between the system Integrator and the Service Provider. Applications based on services have dynamicity as an important factor. The main aim of the proposed approach is to perform selection of test cases during running state of the application which is using the services.

To achieve a safe regression testing technique, some assertions are made on Quality of Service (QoS) and some

scenarios. They did not consider the specification changes in their approach. The toolkit is used to analyse JUNIT Test Suites and then XML encoded test suites are generated out of it [7].

During this phase of research, all the regression testing techniques were mostly code based. Model based regression testing techniques for web services are also proposed where models were created to identify the changes and impact of these changes on the application. Models function to describe the service interface. Finite state automatons are employed for external behavior. For establishing data dependencies, bipartite dependency graph is created. In this graph methods and classes are represented by nodes. Then an algorithm for test case selection is proposed [14].

Bai et al. Researched further into the WSDL Specification. They produced test data on the basis of operations specified in the WSDL specification of the Web Services. They also took into consideration the sequence of operations in the Web Service. WSDL Interfaces were analyzed to generate test data for testing. Their approach does not concentrate on the regression testing of Web Services.

Another study used the WSDL Specification approach to perform regression testing of web service based applications. WSDL specification was used to select the test cases for regression testing [8]. WSDL specification XML is parsed and data type changes are selected. Then, tree is generated out of the parsed XML. Tree is generated for both the original and the changed Web Service. Then, a comparison is made between the trees and test cases are selected using a regression test selector. This selector uses boundary value analysis for identifying the test cases. They have built an automated tool to support their approach.

Masood and Nadeem later extended their approach to include the port type element for identifying the changes in the specification WSDL. Port Type element contains the operation name, input message and output message of the specification WSDL. A tree is created for both the original and changed WSDL. A comparison between the trees is made to select the test cases to be included in the regression test suite.

Our approach takes the port type element from the specification WSDL [8] and then creates a hash table for both the original and changed WSDL. Then, a comparison between the hash tables gives the changes in operation names for the web service. Based upon the operation changes identified, test cases are selected to be included in the regression testing.

III. PROPOSED APPROACH

Software Regression Testing is used to identify unintentional bugs or errors that may have cropped up in the code as a result of changes made in the application software. The ease of developing web services and interdependencies between services and the application increases the pressure on Web Service Testers to ensure that web services are reliable, robust, scalable and secure. Safe Regression Testing is essential in ensuring that web services work as expected throughout the lifecycle of the Application.

Retesting of modified Web Service using regression testing techniques could be very costly as we end up testing large number of test cases. Service Requester does not have the code, so he has to somehow integrate the changed web service in the application and test it thoroughly for a Safe behaviour. Regression testing costs for a web service can be reduced significantly by identifying the changes in the web service and selecting test cases accordingly. This approach avoids the costly creation of new test cases and the useless and not so productive rerunning of existing test cases when it can be guaranteed that the unchanged code of web service will produce the same results as it produced previously.

Our approach utilizes the specification WSDL for identifying the changes in the web service and consequently selecting test cases pertaining to the changed part of the application. WSDL specification contains the web service description and all the details of the operations it supports. Important components of WSDL are: Port Type, Binding, types and messages.

Message: can be input or output. It abstracts the data definition which is being transmitted.

Operation Name: gives the name of the operation implemented by the web service.

Input message: gives the expected abstract input parameter and type information.

Output message: gives the abstract output given by the web service.

Binding Element: provides operation and message details and message format.

Port Type: gives all the operations that are part of the web service. All these operations are performed by the web service.

Type: provides definition of data type. It is a description of the exchanged message being used by the web service.

```
<wsdl:portType name="Service1Soap">
        <wsdl:operation name="AddNumbers">
            <wsdl:input message="tns:AddNumbersSoapIn"/>
            <wsdl:output message="tns:AddNumbersSoapOut"/>
        </wsdl:operation>
        <wsdl:operation name="SubtractNumbers">
            <wsdl:input message="tns:SubtractNumbersSoapIn"/>
            <wsdl:output essage="tns:SubtractNumbersSoapOut"/>
        </wsdl:operation>
</wsdl:portType>
```

Fig. 2: WSDL Port Type

```
<s:element name="AddNumbers">
  <s:complexType>
    <s:sequence>
      <s:element name="number1" type="s:int" maxOccurs="1"
       minOccurs="1"/>
      <s:element name="number2" type="s:int" maxOccurs="1"
       minOccurs="1"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

Fig. 3: WSDL Message

We track the changes in the operation name, element name and type elements of the WSDL. We define the original Web Service as the base line Web Service. The changed web service is termed as Delta Web Service. Our approach has three components XML Parser, comparator and Regression Test Selector.

XML Parser: The function of the XML parser is to parse the WSDL XML for both the baseline and delta versions of the Web Service. XML Parser extracts each element of the WSDL and stores it in a hash table. Operation name of the Port Type element forms the key of the hash table. Each operation forms a unique key for the hash table. All the values related to each operation namely output, parameter count, element name and element type is stored in a list. This list forms the value of each key in the hash table. Every operation key has its own list of values. Algorithm for the XML parser is given in Fig. 5. It works on the principle that each operation name in the web service is unique and can be used as key in the hash table.

Flow of information in the WSDL is as given below [16].

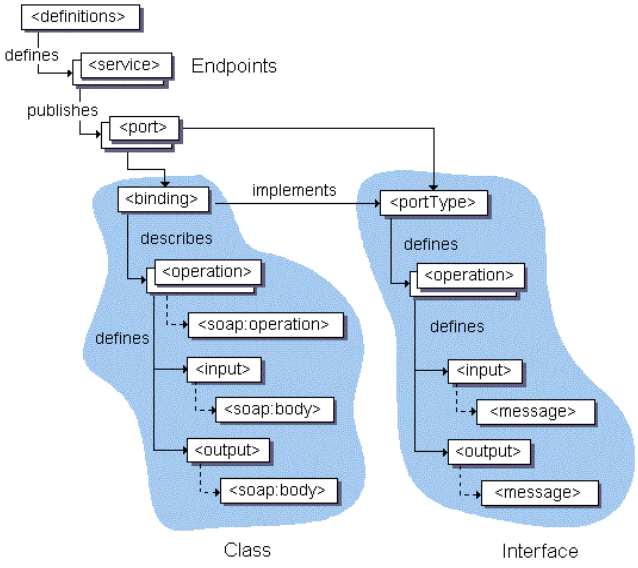

Fig. 4: WSDL Element Relationship

Once the operation name is selected, an empty list is created for storing all the other relevant information regarding the operation: output type, parameter count, input parameter name and input parameter type. This way we are able to store the information regarding each operation in the WSDL into a hash table. XML parser is run both for the baseline and delta WSDL.

Input: WSDL Specification XML

Output: Hash table **H**

Algorithm: Parse WSDL XML and store it in hash table.

1. Select the <PortType> element of the WSDL specification.
2. Select the <operation> element under the <PortType> element. Select the name of the operation and assign it as the key for the hash table <key, value> pair.
   Operation Name is the unique key value for the Hash table.
3. Create an empty List **L** which will store all the values related for a particular Key.
4. Select the <input message> element and count the number of <sequence> elements associated with the <input message>. This count will give the number of parameters for the selected operation name.
5. Add the count to the List **L**.
6. Select the <input message> element and using the input message name extract the parameter names and their respective types for the selected operation name.
7. Add the parameter names and their respective values to the List **L**.
8. Select the <output message> element and using the output message name extract the return type for the selected operation name.
9. Add the return type value to the List **L**.
10. Link the List **L** as the value part for the selected operation name key in the Hash table.
11. Repeat Steps 2-10 for all the <operation> elements in the WSDL.

Fig. 5: Algorithm for XML Parser

Hash table: Generated hash table has operation name as the key and values stored as a list for each key. Format for hash table is shown in Fig. 6.

Key	Value
Operation Name	Return type Parameter count Input parameter name 1 Input parameter name 1 type Input parameter name 2 Input parameter name 2 type . . . Input parameter name N Input parameter name N type

Fig. 6: Hash Table

Comparator: Once we have the hash tables for the baseline and Delta WSDL, we pass them to a comparator, which compares the values based upon operation keys. Comparator first scans the Delta Hash table. For each operation key that also exists in the baseline Hash table, the set of values are compared. In case the values match nothing is done. Case where the values do not match, corresponding operation key is stored in a list. This process is repeated until we have traversed all the keys in the Delta Hash Table.

Input:
Baseline Hash table **H**
Delta Hash table **H'**

Output:
List Of keys from the <key, value> pair with value changes ➔**L1**
List of keys added in the hash table **H'**➔**L2**
List of keys deleted from the hash table **H'**➔**L3**

Algorithm:

1. For each key **opkey** in delta hash table H', Repeat step 2.
2. If hash table **H** contains **opkey**
2.1 Compare the List of values associated with the **opkey.**
2.2 If all the values do not match, add the **opkey** to List **L1**
Else
Add the **opkey** to List **L2**

3. For each key **oldkey** in hash table **H**, Repeat step 4.
4. If hash table **H'** does not contain **oldkey**
4.1 Add the **oldkey** to List **L3**

Fig. 7: XML Comparator

In case, no corresponding match for the operation key is found in the baseline hash table, it is stored in a separate list.

Now, baseline hash table is traversed. We look for any operation name that is not present in the delta hash table. These are the operations which are no longer available in the changed Web Service. Any reference to them should be removed from the application and application be tested thoroughly. All the test cases pertaining to these obsolete operations should be removed or archived from the current testing suite.

Regression Test Selector: We have a baseline testing suite where each test case run is associated with an operation name. Changes in the operation keys from the comparator are used to select, add, modify and remove the test cases from the baseline test suite. This forms our delta test suite. We can have different situations based upon the operation changes.

1. **Change in values related to Operation Key:** All the test cases related to the operation key are reviewed and modified if required. Regression testing is done to ensure that the web service behaviour is the same as was before the changes.
2. **No change in Operation Key:** Test Cases related to the operation key may not be run, and included as it is in the delta test suite.
3. **Operation Key Added:** New Test Cases have to be created for the operation key and tested thoroughly as per the expected behaviour. These test cases will also be included in the delta test suite.
4. **Operation Key deleted:** All the test cases related to the operation key are removed from the test suite. As the web service no longer performs the operation, application will need to be changed and tested for any references to the obsolete operation.

Now, Regression Testing can be performed based on the delta test suite. We already have the list of operation name changes and each test run is associated with the operation name. We can selectively run only those test cases which are associated with the modified operations in the web service. This will ensure a complete and efficient regression testing of the system.

This enlists a comprehensive approach towards maintaining and selecting a regression test suite.

IV. CONCLUSION

In this paper we presented a specification based test case selection for regression testing of web services. Our approach uses the specification WSDL and parses its elements into a hash table. Research has shown that, in some situations, hash tables are more efficient and fast than search trees or any other table lookup structure like dictionary.

Time Complexity in Big O Notation	Hash Table		Binary Tree	
	Average	Worst	Average	Worst
Space	O(n)	O(n)	O(n)	O(n)
Search	O(1)	O(n)	O(log n)	O(n)
Insert	O(1)	O(n)	O(log n)	O(n)
Delete	O(1)	O(n)	O(log n)	O(n)

Fig. 8 Time complexity of Hash table and Binary Tree

We have extended the approach of Masood and Nadeem and used hash table to store the WSDL XML instead of an operation tree. Use of hash table makes the comparator algorithm much simpler. Also, based on the time complexity hash table on an average perform better than search trees. The regression test selector uses the operation changes identified by the comparator. This forms the basis of selecting the test cases for the delta web service from the baseline web service. Hence, provides a safe regression test selection approach.

V. FUTURE RESEARCH

In future we plan to enhance our work by automating the test generation and selection process for regression testing based on the output from the comparator. Our research is based on the operation name as the key for the hash table assuming each operation to be unique. We can consider using a composite key for the hash table for increasing the efficiency of the hash table.

REFERENCES

[1] K. Gottschalk, H. Kreger, J. Snell and S. Graham, "Introduction to Web Services Architecture", IBM Systems Journal, NO 2, VOL 41, pp. 170-177, (2002)

[2] Simple Object Access Protocol (SOAP) 1.2, Part 2, "World Wide Web Consortium" - Adjuncts: (2007), http://www.w3.org/TR/soapl 2-partO/

[3] Web Services Description Language (WSDL) 2.0, part 1: "World Wide Web Consortium" - Core Language (2007) , http://www.w3c.org/TR/wsdl20/

[4] . Ha and H-S. Park, "QoS based Client Information for Semantic Web Service" - International Journal of Software Engineering and Applications, Vol. 3, No. 1 (2009)

[5] M-H. Lee, C-J.Yoo and O-B. Jang, "Embedded System Software Testing Based on SOA for Mobile Service" - International Journal of Advanced Science and Technology, Vol.1 (2008)

[6] R. Binder, "Testing Object-Oriented Systems: models, patterns and Tools", ISBN-10: 0321700678, Addison-Wesley Professional; Edition 1, ISBN-13: 978-0321700674 (2000)

[7] M. D. Penta, G. Esposito, M. Bruno, G. Canfora and V. Mazza, "Web Services Regression Testing and Test and Analysis of Web Services", Barresi, L., Nitto, E.D. (eds.) "Test and Analysis of web Services", pp. 205-234. Springer, Heidelberg (2007)

[8] T. Masood, A. Nadeem and G. S. Lee, "A safe Regression Testing Technique Based on WSDL Specification" - Software Engineering Business Continuity and Education Communications in Computer and Information Science, Springer Berlin, Heidelberg (2011)

[9] T. Masood, A. Nadeem and S. Ali, "An automated approach to regression testing of web Services based on WSDL operation changes" – Emerging Technologies(ICET), IEEE 9th International Conference 2013

[10] XML Schemas - Part 2, "Data types - World Wide Web Consortium", http://www.w3.org/TR/xmlschema-2/

[11] P. C. Jorgensen, "Software Testing: A Craftsman's Approach", second ed., LLC, CRC Press, 2002.

[12] M. Ruth, F. Lin and S. Tu., "Applying Safe Regression Test Selection Techniques to Java Web Services", 1, 10 - (2006)

[13] M. Ruth, S. Tu, S. Oh, A. Loup, B. Horton, O. Gallet and M. Mata, "Towards Automatic Regression Test Selection for Web Services", 31st Annual International Computer Software and Applications Conference, July 2007 24-27, Beijing, China.

[14] T. A. Khan and R. Heckel., "A Methodology for Model based Regression Testing of Web Services", Academic and industrial Conference - Practice and Research Techniques, pp. 123-124, (2009)

[15] X Bai, W Dong, WT Tsai, Y Chen, "WSDL-based Automatic Test Case Generation for Web Services Testing", IEEE International Workshop on Service-Oriented System Engineering (SOSE), Planned Parenthood.215-220. IEEE Computer Society, (2005), Los Alamitos

[16] http://download.oracle.com/otn_hosted_doc/jdeveloper/1012/web_services/ws_wsdlstructure.html