Web Service Test Case Reconfiguration Framework for Automatic Adaption to Changes

Sirisha K L S, Dr. M. Chandra Mohan

Research Scholar, Department of Computer Science and Engineering, JNTUH.
Professor , Department of Computer Science and Engineering, JNTUH

klssirisha@gmail.com, c_miryala@jntuh.ac.in

Abstract-Web services paved way for realizing component technology that helps in reusing software components with inter-operability. Such technology helps in developing applications that are heterogeneous in nature and yet have seamless integration among them. Modern enterprise applications are widely using this technology to have business collaborations. Thus it reaps benefits of reusability and integration with ease. Enterprise applications can be built with in-house web services and also third party web services. This has led to development of third party web services API (Application Programming Interface). Testing web services in distributed environment is a challenging problem. When business logic of web services is changed, it is essential to rewrite test cases or modify them. In large scale applications with plenty of business processes, it is a tedious task. To overcome this problem, this paper proposes a framework that automates reconfiguration of test cases when web service source is modified. The framework has provision to determine the kind of change occurred and automate the process of reconfiguration accordingly.

Keywords: Distributed computing, web services, testing, test case reconfiguration, change dynamics

1. INTRODUCTION

Distributed computing technology enabled the development of web services technology on top of it. The web services technology has two distinct advantages such as ability to create reusable and interoperable software components and integration of heterogeneous applications irrespective of their development platforms. When web services are located in different servers geographically located in different places, it is not easier to test those applications. When test cases are written, they are used for testing. However, the web service functionalities might change in future. When the changes are not reflected in the test case functionality, it needs

to be updated manually. Therefore, to overcome this problem, this paper has provided the framework for automatic reconfiguration of web service test cases.
There are different phases in the development process. First of all, a web service needs to be created and it has to be registered with UDDI registry. Or it can be deployed in a server like Apache Axis server. Once web service is deployed, it is possible to use it by other parties with the help of WSDL file which describes about web service and how to reach it. After deployment, the web service may be subjected to different changes like change in arguments of a specific function, adding new functions or removing existing functions. These changes are to be detected in order to have automatic reconfiguration of test case of the web service in question.
The proposed framework has provision for change notifications and trigger test case reconfiguration automatically. Once the test case is subjected to changes in source code, its WSDL also reflects the same. Such WSDL is used to know the changes and make automatic reconfiguration of test cases. Our contributions in this paper are as follows.

1. A framework is proposed for automatic reconfiguration of test cases associated to web services.
2. A porotype application is built using Java programming language to show the reconfiguration process.
3. Evaluation is made and found that the framework is useful in automatic test case reconfiguration.

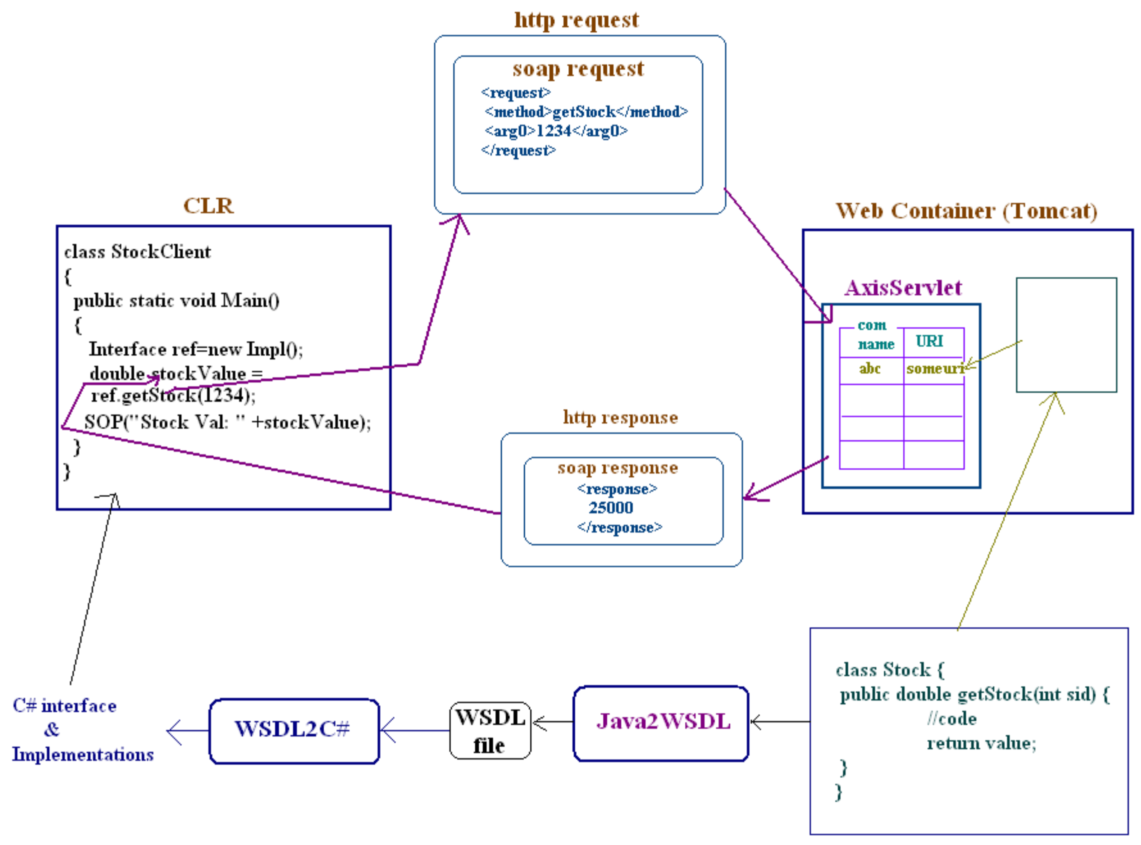
The remainder of the paper is structured as follows. Section 2 provides review of literature related to web services testing. Section 3 presents the proposed framework and its implementation Section 4 concludes the paper and provides directions for future work.

2. RELATED WORK

This section throws light into the review of literature on web services and testing them. Li et al. [1] focused on analysing web services in terms of control flow analysis and coverage driven testing. Morales et al. [2] on the other hand investigated web services for their integrity with passive testing via timed invariant testing. Yu et al. [3] reviewed different approaches in testing web services while Siblini and Mansour [4] performed different kinds of testing. Cooray et a. [5] specifically concentrated on the test reconfigurations for composite web services. Chandrasekaran et al. [6] and Senthilin et al. [9] focused on the performance analysisof web services. With respect to service oriented applications, Cooray et al. [7] test reconfiguration possibilities are explored.
With respect to large scale service networks, Oh et al. [8] investigated on the composition and effectiveness of web services. Bucchiarone et al. [10] studied the testing of service composition and the way of understanding test cases. Tsai et al. [11] investigated on the deployed web services that are registered and accessed through UDDI. Bai et al. [12] worked out on the concept of dependence-based progressive group testing. The concept of safe regression test is carried out on web services by Lin et al. [13] while robustness is testing by Barbara et al. [14].Hekell and Lohmann [15] focused on the contrast based web service testing.More recent work is found in [16], [17], [18], [19] and [20]. From the literature, it is understood that there is need for reconfiguration of web services in the context of its usage widely in the real world. When test cases are not reconfigured, the testing process becomes tedious task. This paper focused on the proposal of a framework for dynamic reconfiguration of test cases.

3. WEB SERVICE TEST CASE RECONFIGURATION FRAMEWORK

A framework is proposed to realize automatic reconfiguration of web service test cases. Web service architecture is shown in Figure 1. It is based on the broker architecture. It has different standards associated with web services. They are known as Simple Object Access Protocol (SOAP), Universal Description Discovery and Integration (UDDI) and Web Service Description Language (WSDL). When web service is created, a WSDL file is required in order to provide details of web service. This file is registered with the UDDI registry. A client application needs to make a lookup into the registry in order to find the details of web service needed. Once the details are known, the client application uses the reference returned by the UDDI registry to avail the functionalities of web service.



```
<request>
 <method>getStock</method>
 <arg0>1234</arg0>
</request>
```

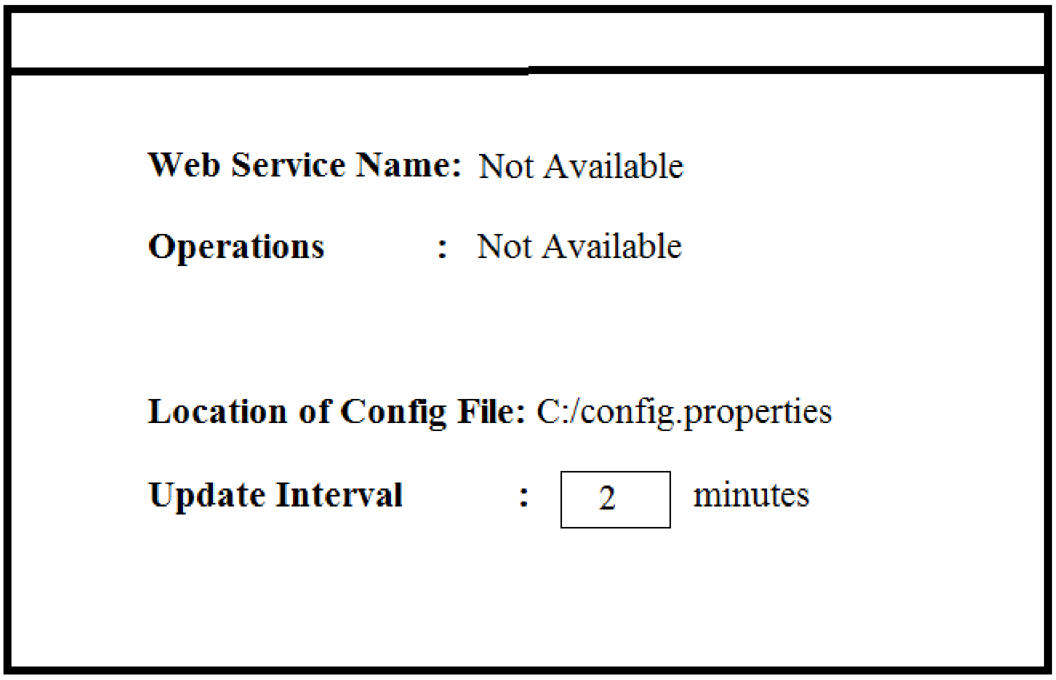
```
class StockClient
{
 public static void Main()
 {
   Interface ref=new Impl();
   double stockValue =
    ref.getStock(1234);
   SOP("Stock Val: " +stockValue);
 }
}
```

```
<response>
25000
</response>
```

```
class Stock {
 public double getStock(int sid) {
          //code
          return value;
 }
}
```

Figure 1: Web services architecture and flow in developing and consuming services

As presented in Figure 1, the web service development is made using any object oriented programming language. Once web service is developed, its WSDL file is generated. Then the WSDL file along with web service details are registered with UDDI. Consumer of web service need to know the WSDL file in order to ascertain the service and write client program. The web service is kept in server and it is identified by the SOAP end point. In case of Apache Axis Server, the SOAP end point is known as AxisServlet. The web service is developed in Java and it is consumed using C# program as illustrated in Figure 1. A testing framework we proposed is presented in Figure 2.



Web Service Name: Not Available

Operations : Not Available

Location of Config File: C:/config.properties

Update Interval : 2 minutes

Figure 2:A part of the user interface of the proposed framework

As presented in Figure 2, the initial screen of the proposed framework shows the web service name and operations. When web service is not chosen, they are shown as “not available”. It also shows the location of configuration file which holds key value pairs that are used to know the details of web service being tested for reconfiguration. The time interval is used to trigger the reconfiguration functionality automatically. The test case created for given web service is reconfigured automatically based on the changes made to web service source.

3.1 Sample Web Service

```
package com.ws.server;


import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;


@WebService(serviceName = "CalcWebService")
public class CalcWebService {


   @WebMethod(operationName = "add")
   public int add(@WebParam(name = "arg1") int x, @WebParam(name="arg2") int y) {
      return x+y;
   }
   @WebMethod(operationName = "sub")
   public int sub(@WebParam(name = "arg1") int x, @WebParam(name="arg2") int y) {
      return x-y;
   }


 @WebMethod(operationName = "mul")
   public int mul(@WebParam(name = "arg1") int x, @WebParam(name="arg2") int y) {
      return x*y;
   }


 @WebMethod(operationName = "div")
   public int div(@WebParam(name = "arg1") int x, @WebParam(name="arg2") int y) {
      return x/y;
   }
}
```

Listing 1: Same web service

3.2 Sample Java Code for Discovering Web Service Detail from WSDL

```
public class Calculator {
  public static void main(String argv[]) throws XPathExpressionException {
    try {
            File  fXmlFile  =  new  File("C:\\Users\\sankalpa\\Desktop\\WebService\\web\\WEB-
INF\\CalWebService.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(fXmlFile);
            doc.getDocumentElement().normalize();
         XPath xPath =  XPathFactory.newInstance().newXPath();
          String expression = "/definitions/operation";
            System.out.println("Root element :" +
doc.getDocumentElement().getAttribute("name"));
            NodeList nList = doc.getElementsByTagName("operation");
            System.out.println(nList.getLength());
        Node n=null;
      // Element eElement=null;
        for (int i = 0; i < nList.getLength(); i++) {
            Node nNode=nList.item(i);
           if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                        Element eElement = (Element) nNode;
                        System.out.println(eElement.getAttribute("name"));
           }
            System.out.println("\nCurrent Element :" + nNode.getNodeName());
        }
    } catch (Exception e) {
            e.printStackTrace();
    }
  }
}
```

Listing 2: Discovery of web service

3.3 Simple Test Case with JUnit

```
import com.ws.server.CalcWebService
import com.ws.server.CalcWebService;
import org.junit.Test;
import static org.junit.Assert.*;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class CalculatorTest{
 CalcWebService c = new CalcWebService();
 @Test
 public void testAdd() {
assertEquals(22, c.add(12,10));
}
 @Test
public void test Sub() {
 assertEquals(15, c.sub(20,5));
}
 @Test
 public void test Mul() {
 assertEquals(50, c.mul(10,5));
 }
 @Test
 public void testdiv() {
 assertEquals(10, c.div(100,10));
 }
public static void main(String args[]) {
  Result result = JUnitCore.runClasses(CalculatorTest.class);
 for(Failure failure : result.getFailures()) {
 System.out.println(failure.toString());
  System.out.println(result.wasSuccessful());
}
}
}
```

Listing 3: Simple JUnit Test for Testing Web Service

3.4 Setting Timer and Periodic Detection of Changes in Web Service

```
public void start() {
         delay=Integer.parseInt(jTextField1.getText())*1000;
        timer.cancel();
        timer = new Timer("TaskName");
        Date executionDate = new Date(); // no params = now
        timer.scheduleAtFixedRate(task, executionDate, delay);
    }
    private class LoopTask extends TimerTask {
        public void run() {
           NewClass newclass=new NewClass();
            newclass.main(null);
        }
    }
```

Listing 4: Determining the kind of change made to web service

As illustrated in Figure 2, the source code provided in Listing 4 is used to determine the kind of change made to web service. A simple web service source is provided in Listing 1 while Listing 2 makes use of WSDL file to discovery the details of the existing web service. The Listing 3 on the other hand provides a test case associated with the simple web service. When changes are made to web service, the source code shown in Listing 4 will find the change and it triggers modifications in the test case automatically.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a framework for automatic reconfiguration of test cases associated with web services. Since web services may be subjected to changes in business logic and different number of functions, it is desirable to have automatic reconfiguration of test cases when web service source is modified. Towards this end, the proposed framework performs the reconfiguration in two phases. In the first phase, it determines the kind of change made and in the second phase, it actually adapts the changes by reconfiguration of test cases automatically. The web service for which test cases are reconfigured may be a local web service or remote web services. Web Service Description Language (WSDL) plays crucial role in the process of identification of change made and also adapt to changes by modifying the test case accordingly. An empirical study is made to know the utility of the proposed system. The experimental results revealed that the proposed framework is useful in the real world. In future, we intend to improve the framework for complex web services in different domains.

References

[1] Li Li, Wu Chou, WeipingGuo. (2008). Control Flow Analysis and Coverage Driven Testing for Web Services. *IEEE*, p271-280.

[2] Gerardo Morales, Stephane Maag, Ana Cavalli. (2010). Timed Extended Invariants for the Passive Testing of Web Services .*IEEE*, p.20- 30.

[3] Lian Yu, Wei-Tek Tsai1, Xiangji Chen, Linqing Liu, Yan Zhao, LiangjieTang,and Wei Zhao2 . (2010). Testing as a Service over Cloud . IEEE, p1-8.

[4] Reda Siblini, Nashat Mansour. (2005). Testing Web Services. *IEEE*, p1370-1381.

[5] Mark B. Cooray, James H. Hamlyn-Harris and Robert G. Merkel . (2013). Dynamic TestReconfiguration for Composite Web Services . IEEE, p1-13

[6] SenthilanandChandrasekaran , John A. Miller, Gregory S. Silver , BudakArpinar& Amit P. Sheth. (2010). Performance Analysis and Simulation of Composite Web Services. IEEE, p1-14.

[7] M. B. Cooray, J. H. Hamlyn-Harris, and R. G. Merkel, "Test recon- figuration for service oriented applications," in Proc. IEEE Int. Conf. Utility Cloud Comput., 2011, pp.300–305.

[8] S. Chan Oh, DongwonLee,Soundar R.T. Kumara. (2008). Effective Web Service Composition in Diverse and Large-Scale Service Networks. *IEEE*. 1 (1),p.20-30.

[9] SenthilanandChandrasekaran , John A. Miller, Gregory S. Silver , BudakArpinar& Amit P. Sheth. (2010). Performance Analysis and Simulation of Composite Web Services. IEEE, p1-14.

[10] A. Bucchiarone, H. Melgratti, and F. Severoni, "Testing service composition," in Proc. 8th Argentine Symp. Softw. Eng., Mar del Plata, Argentina, 2007, pp. 1–16.

[11] W. T. Tsai, R. Paul, Z. Cao, L. Yu, and A. Saimi, "Verification of web services using an enhanced UDDI server," in Proc. 8th Int. Workshop Object-Oriented Real-Time Dependable Syst., 2003, pp. 131– 138.

[12] X. Bai, Z. Cao, and Y. Chen, "Design of a trustworthy service broker and dependence- based progressive group testing," Int. J. Simul. Process Modell., vol. 3, pp. 66–79,2007.

[13] F. Lin, Michael Ruth, Shengru Tu. (2006). Applying Safe Regression Test Selection Techniques to Java Web Services. *IEEE*, p1- 10.

[14] C. Barbara G. Ryder,AnaMilanova,DavidWonnacott. (2004). Testing of Java Web Services for Robustness. ACM,p271-280

[15] Reiko Heckel,MarcLohmann. (2005). Towards Contract-based Testing of Web Services. *Elsevier*. 116, p1-10.

[16] AnshuSoni, VirenderRanga. (2019). API Features Individualizing of Web Services: REST and SOAP. *International Journal of Innovative Technology and Exploring Engineering*. 8 , p664-671.

[17] Xuetao Tian, Honghui Li and Feng Liu. (2017). Web Service Reliability Test Method Based on Log Analysis. *IEEE International Conference on Software Quality, Reliability and Security* , p-195-199

[18] Huwenhi,Huangyu,lixxueyang,Xu Chen. (2017). Study on REST API Test Model Supporting Web Service Integration. *IEEE 3rd International Conference on Big Data Security on Cloud*, p133-138.

[19] GulshanSaleem, FarooqueAzam,Muhammad U sman Y ounu,Nisar Ahmed and Li Yong . (2016). Quality Assurance of Web Services: A Systematic Literature Review. *IEEE* Review. *IEEE International Conference on Computer and Communications*, p1391-1396.

[20] SitiRochimah and AlhajiShekuSankoh,. (2016). Migration of Existing or Legacy Software Systems into Web Service-based Architectures (Reengineering Process): A Systematic Literature Review. *International Journal of Computer Applications* .113 , p0975 – 8887.