International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 22619-22626

http://www.ripublication.com

Regression Testing on Services In Mobile Applications

Rajkumar J. Bhojan[1],K. Vivekanandan[2], Pankaj Moses Monickaraj[3], Subramanian Ganesan[4],

[1.] *Principal Consultant, Wipro Technologies, Bangalore, India. jbrkumar@gmail.com*
[2.] *Professor, BSMED, Bharathiar University, India. vivekbsmed@gmail.com*
[3.] *Assistant Professor, Christ University, Bangalore, India. pankj.moses@gmail.com*
[4.] *Professor, Electrical & Computer Engineering, Oakland University, Rochester, MI, USA.*ganesan@oakland.edu

Abstract

The power of mobile devices has increased dramatically in the last few years. The Mobile apps market increases every day and Mobile device has become one of the most important equipment in people's daily life, which brings us not only convenience of communication, but more and more work and entertainment applications. Mobile testing becomes very crucial as the mobile applications and mobile users are growing rapidly. Test consultants, test specialist, test managers and software engineering researchers are finding ways to do effective verification methods and to ensure reliability of mobile applications. In this paper, we propose regression testing framework on services in Remote Link Lite (RLL) mobile application. In order to perform regression test on mobile application, we have considered RLL application for POC purpose;

Keywords: Mobile Test Automation, Web Services, Automotive testing, Framework, Regression Testing

Introduction

In[1], a web service refers to a programmable Web application with a standard interface and is universally accessible using standard network protocols. Its interoperability provides users a practical way to quickly create a new service by combining existing services. It is one central reason why this emerging technology has caught significant momentum from both academiaand industry in the recent years.

Web Service Testing

Presently, web services are having an increasingly importantrole in the construction of computer applicationsused remotely by many users. It is not possible for a

Webservice provider, to make available a new Web service orto make a radical change from an old Web service withoutfirst checking this service. As the software system, a webservice (WS) can be tested by traditional software testingtechniques such as: conformance testing, performancetesting, availability testing, robustness testing, etc. Webservices are applications that allow the formation of SOA(Service-Oriented Architectures) applications which are builtby composing other web services.

As authors in [2] say, testing web service compositions is more complex than testing conventional software systems due to various problems that can be found at a service, orchestration and infrastructure levels. Its implementation requires orchestration of web services that are built and deployed on heterogeneous infrastructures and platforms. Sometimes these services wrap the functionality of legacy systems and components that are outside the organizations' boundaries and therefore are very hard to be tested. Furthermore, they could be unavailable for a given period of time or in the worst case could be un-deployed by their provider. This in turn leads the necessity of emulation of the missing web services. [2]

The model of Web services poses significant challengeson mobile application testing due to its unique nature and properties. As per authors in [3],First, unlike screen-centric Web applications,Web services areWeb components only accessible via interfaces published in standardWeb services-specific interface definition languages (e.g., WSDL) and accessible via standard network protocols (e.g., SOAP). Therefore, how to design test cases for a Webservice using its limited information exposed remains a challenge. Second, one major goalof adopting the Web services technology is to dynamically compose existing Web servicesas components to quickly construct a new service. Therefore, Web services testing usuallyimply that multiple service components have to be tested in an efficient and lightweight (i.e.,without maintaining dedicated network connections throughout a testing process) manner.Third, how to mitigate the overhead caused by theWeb services-specific transport protocols(e.g., SOAP) deserves special attention [3].

Mobile Service Testing

Authors in [4] indicate that mobile apps are everywhere, business transactions are being entertained by some apps. Authors gave an example asmost of the banking apps on a mobile devices act as a front end that invoke services on a back-end server of the bank, which might contact even more servers. Klaus Haller [4] reiterates that the essence of mobile test automation. There are two reasons to automate mobile test cases: to ensure minimalfunctional coverage and to achieve scalable test configuration coverage.Minimal functional coverage is a safety net. The test scripts cover thebasic features of the app and run, for example, each night. They coverfront-end tests and front-end–back-end-integration tests. A basic testinfrastructure consists of a PC running the test script, one local device,and an automation tool. Scalable test configuration coverage checks whether the app runsproblem-free on all relevant devices and OS versions. Test scripts mustrun in parallel on multiple devices, which require a private devicecloud with parallelization features. Besides, since most of the customers demand a flawless experience, we must test mobile apps and websites on the actual devices that they are using - at least the most popular ones representing

each OS. Based on market research and internal analytics, the devices should be chosen for regression testing.

Mobile Testing challenges

Authors B. Kirubakaran, et al., listed the following mobile testing challenges in their paper "Mobile Application Testing - Challenges and Solution Approach through Automation".

1. As mobile apps receive inputs from different providers like users, sensors and connectivity devices, it leads to the unpredictability and high variability of the inputs.
2. Languages add some specific constructs for managing mobility, energy consumption and sensing.
3. In order to do mobile apps functional testing, it needs application and the environment like airplane mode, meeting and low battery.
4. In GUI testing, the real challenge lies in testing native applications on devices and adequate rendering of data by different devices.
5. Mobile device performance and reliability testing depends on the mobile resources, operational mode and connectivity quality.
6. Security testing is a major challenge as mobility of the device into networks with different security levels. [5]

The following are the criteria for choosing mobile devices

As mentioned in [6] and [7], the following are the list of factors to be considered when choosing set of devices for mobile testing.

- Top supported devices and manufacturers
- Target market (age, geography, business/pleasure/youth, etc…)
- Supporting devices : Smartphones, tablets or both
- Relevant screen sizes
- Relevant regions, carriers and network technologies
- Major supporting OS (i.e., iOS, Windows, Android) [6][7]

Existing Framework

According to authors in [8], Automated testing of distributed, service-oriented applications generally falls into two execution models: centralized and decentralized. In a centralized model, a testing controlservice sends testing commands to services installed on each host taking part in testing, generally in a push model. In a decentralized model, there is no centralized control service, and testing occurs in a purely distributed manner. If synchronization is required, decentralized infrastructures may use network lock files, or they may require separate daemons with intimate knowledge of test sequencing orcustom messaging protocols. In this framework, web services are dealt with mobile applications and web application servers. Unlike the above mentioned framework, our framework is dealing with mobile applications with real vehicles communications.

In [9], an approach is proposed to generate a testbed for service-oriented systems that takes into account a mobility model of the nodes of the network in which the

accessed services are deployed. According to authors [9], this testbed allows a tester to assess off-line the QoS properties of a service under test, by considering possible variations in the response of the interacting services due to node mobility.

In [10], author has introduced AppACTS framework. It is aimed at helping developers to improve mobile application compatibility testing efficiency, save cost and ensure mobile application quality and reliability. Mobile App Automated Compatibility Testing Service (AppACTS) is an online mobile app compatibility testing cloud facility from where end users could upload a mobile app for automated device compatibility testing and view the testing result.AppACTS is based on a scalable architecture that supports to plug in and pull out mobile devices on demand and add or reduced slave nodes and mobile servers at any time.

Communication Between Mobile Devices And Vehicles

According to author in [11], the last two decades have seen computer and communication technology become more prevalent in cars, enabling the incorporation of systems and devices for both driver support (e.g., navigation aids) and infotainment (e.g., news and email) [11]. Most of the major car manufactures like GM, Ford, Nissan, Lexus, Audi, etc., use mobile applications as enhanced features in their cars. The mobile applications such as RemoteLink [12], ApplinkSync [13], Audi Connect [14] and myBMW [15] are communicating with owner's cars and control the vehicles from anywhere.RLL is GM OnStar's mobile application which is used for making communication between user and vehicle. The communication between user and the vehicles are based on service requests and responses. This paper is focusing on how our service based test automation framework will help in executing these type complex automotive test cases.

The following are some of the generic operations of the above said mobile applications

a) Makes a secure connection between vehicle and mobile device.
b) Start and stop vehicle engines
c) Check real-time fuel and oil levels
d) Hands free calling
e) Lock and Unlock cars
f) Navigation set up
g) Vehicle Finder

In order to perform the above mentioned operations through mobile application, we propose a test automation framework for evaluating mobile application's operations whenever they communicate with corresponding vehicles. The steps involved in mobile vehicle communication are described in the following algorithm.

Table 1. Algorithm for Service Connection

1. Start
2. Make a connection with Vehicle
3. Send Lock Command
4. If Request = Successful,
   Successful Message from the Vehicle "Vehicle is Locked"
   Else
   Throw Connection Error Unsuccessful Message from the Vehicle
5. End

As shown in the Table 1, the authorization server validates the request. If the request is valid, the authorization server authenticates the user and obtains the authorization decision from the user. If the user grants the access request, the authorization server issues an authorization code and delivers it to the client.

Implementation:

As a normal procedure in Services, we register our application with the service we want to access and get some unique identifiers for our application. When user wants to access something on the service, they make a request to the service using the unique identifier and telling it where to send after they authenticate. User will login to the service using the mobile app launched for that service and choose to grant us application certain privileges.The service will redirect the user to the callback URL we provided and include a code we can use to get an access token. Then we call the service one more time passing in our unique identifier and the code we got back and the service grants us an access token which we can use to access services we have been granted permissions for. The next time we need to make a call we can just use that access token instead of going through this whole process again.

As shown in the Figure-1, ant build will invoke Java Driver class. As authors K.Vivekanandan, Rajkumar Bhojan, S. Ganesan describe in [16], the Java Driver class opens the application, based on the Run Flag status (RUN/NO RUN) given in the data sheet, fetches the data from Data Table and sends it to the AUT (Application Under Test). For better usability, we initiated our service based automated testing from a functional operation, i.e. functional test script will login to mobile app and clicks on "Lock" command button on the mobile device. Lock is one of operations to lock the vehicle doors. This request will be sent to vehicle and the response will be received after proper authentication as shown in the screen shots in Figure 3. Once the first cycles gets completed, the controller goes to next record in the Data Table, it searches for Run Flag Status, if it is "RUN", it executes the second test scripts and so on until it reaches the "END". If run flag status is "NO RUN", controller skips the corresponding test script and moves to the next script. Fig.3 shows the screen shot of lock operation. While the execution was going on, we took the screen shot of the service from the seetest's emulator. [17]

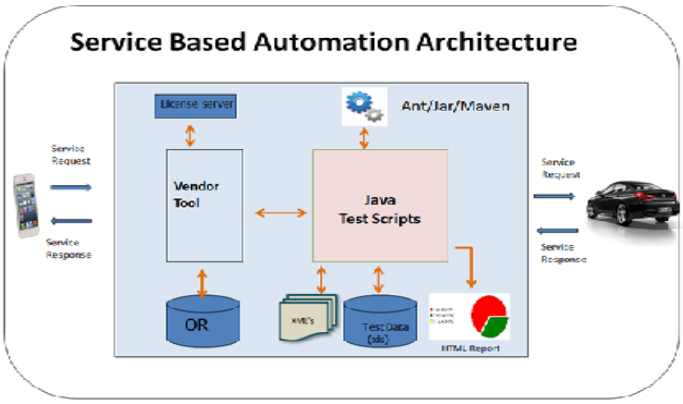

Figure 1: Service Based Test Automation Framework Architecture

In this Service based framework, we used Experitest's Seetest [18] tool for front end operations of the application. With the help of SeeTest object identifier, we extracted the application elements that we wanted to run a test and stored in object repository. Once it was extracted, the elements were showing up in the object repository (OR). Then we incorporate elements to our TestNg framework[17] [19] written in Java. Java scripts were customized for adding features like fetching data from xls sheet and xml files, merging one or more executable methods and sending the reports in html/xslt formats. In this report, we will be able to find out how tests are passed and how tests are failed in a test suite. A sample report table is shown in Table 2. The failed test steps will also have links which redirects to screen shots of the errors captured during the execution. If the request is valid, the authorization server authenticates the user and sends success response to the user. If authentication fails, user will get failed messages from the server.

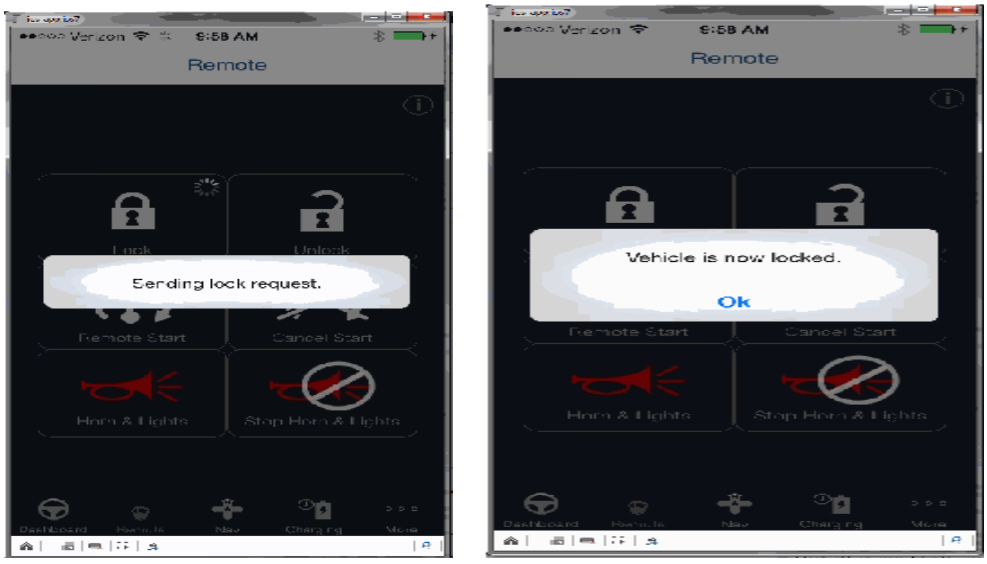

Figure 2: Request and Response from a Vehicle in RLL application

Table 2: Sample HTML output for Lock Command Response

Step #	Step Description	Expected Result	Actual Result	Status
1.0	Login into Remote Link	User must be logged in	User is logged in	Pass
1.1	verify if sending lock message appears	Sending lock message must appear	Sending message appears	Pass
1.2	verify if 'vehicle is now lockeed' alert message is present	Alert must be present	Alert is present	Pass
1.3	verify if 'vehicle is now locked' alert is dismissed on clicking ok	Alert must be dismissed	Alert is dismissed	Pass
1.4	verify if lock icon shows correct update date and time	Lock icon must show correct date and time	Lock icon shows correct date and time	Pass
2.0	Verify if logout is successful	user must return to login page	user returns to login page	Pass

Conclusion

In this paper, we summarize the techniques and mechanisms used increating a test automation framework and analyze how they canlead to an efficient test on mobile applications. We have also studied how automotive software behaved on mobile devices and how to handle and utilizesystem event and Service based testing.We also believe thatthe above said framework can easily be extended to find a broader range of testing in automotive mobile applications.Our future work includes cloud based mobile testing of the WS transactions standardsand their performance evaluation.

References

[1] Jia Zhang, “A Mobile Agent-Based Tool Supporting Web ServicesTesting”, Springer Science Business Media, LLC. 2010.

[2] DenitsaManova, Sylvia Ilieva, DessislavaPetrova-Antonova,” Testing Web Service’s Compositions Following TASSA Methodology”, International Conference on Computer Systems and Technologies - CompSysTech’13 , ACM 978-1-4503-2021-4/13/06

[3] Werner, C., Buschmann, C., & Fischer, S. (2005). WSDL-driven SOAP compression. InternationalJournal of Web Services Research, 2(1), 14–35.

[4] Klaus Haller, “Mobile Testing” ACM SIGSOFT Software Engineering, DOI: 10.1145/2532780.2532813

[5] B. Kirubakaran, Dr. V. Karthikeyani, "Mobile Application Testing - Challenges and Solution Approach through Automation", International Conference on Pattern Recognition, Informatics and Mobile Engineering, Feb 2013

[6] www.perfectomobile.com

[7] http://www.mbweek.com/2013/07/03/3731

[8] James Edmondson, Aniruddha Gokhale, Sandeep Neema, “Automating Testing of Service-oriented Mobile Applications with Distributed

Knowledge and Reasoning", IEEE International Conference on Service-Oriented Computing and Applications, 2011.

[9] Bertolino, A., De Angelis, G., Lonetti, F., Sabetta, A.: Let The Puppets Move! Automated Testbed Generation for Service-oriented Mobile Applications. IEEE, Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference

[10] Jun-fei Huang, "AppACTS:Mobile App Automated Compatibility Testing Service", 2nd IEEE International Conference on Mobile Cloud Computing, Services and Engineering, 2014, DOI 10.1109/MobileCloud.2014.13

[11] Ainojie Alexander Irune, "Evalulating the Visual Demand of In-Vehicle Information System: The Development of a New Method", DOI: 10.4018/978-1-4666-2068-1.ch001

[12] www.onstar.com

[13] http://support.ford.com/sync-technology/applink-overview-sync

[14] http://www.audiusa.com/innovation/intelligence/audi-connect

[15] http://www.bmw.com/com/en/owners/bmw_apps_2013/apps/my_bmw_remote_app/index.html

[16] K.Vivekanandan, Rajkumar Bhojan, SubramaniamGanesan, "Cloud Enabled Test Evaluation on Mobile Web Applications", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 6, June 2014

[17] Rajkumar Bhojan, K.Vivekanandan, "Agent based Test Automation in Mobile-VehicleCommunication", International Journal of Applied Engineering ResearchISSN 0973-4562 Volume 9, Number 21 (2014) pp. 11469-11486

[18] http://www.experitest.com

[19] http://testng.org/doc/index.html