Customising Android Automated Testing Framework To Enable Native Hardware And Software Support

[1]Mrs. Manju Geogy, [2]Dr. Andhe Dharani

[1]*Assistant Professor, Dept of MCA, R.V.College of Engineering, Mysore Road, Bangalore*

[2]*Associate Professor, Dept. of MCA, R.V.C.E, Mysore Road, Bangalore-59*

Abstract

Android development tools and frameworks, as well as, cost-effective testing techniques are becoming essential to guarantee the development of secure, high-quality Android applications, which are the necessity of today's world. This paper deals with the difficulty of automatic testing of mobile applications developed for the Google Android platform, and presents a technique for quick crash testing and regression testing of Android applications. The proposed technique is based on a crawler that automatically builds a model of the application GUI and obtains test cases that can be automatically executed. The technique is supported by a tool for both crawling the application and generating the test cases. This paper presents an example of using the technique and the tool, for testing a real small size Android application that preliminary shows the effectiveness and usability of the projected testing approach.

Keywords **:** Android, Automatic Testing, Robotium, JUnit.

IJERT

I. INTRODUCTION

Android Bridging the gap between desktop computers and hand-held devices is the main challenge that research in mobile applications is addressing for the next future. According to Andy Rubin, Guru for Google's Android, "There should be nothing that users can access on their desktop that they can't access on their cell phone" [1]

Android is an open source Linux based operating system designed specifically for smartphones and tablet computers. It is considered as a software stack [3] for mobile devices as it includes all the key factors for the mobile devices - an operating system, middleware and key applications. It comes with an SDK which provides the tools and APIs necessary to develop new applications for the platform in Java. Android does not distinguish between its core applications and new applications developed with the SDK; all applications can potentially interact with the underlying mobile devices and shared their functionality with other applications.

If android application testing has to be done on the different machines on same time, automation of testing is the best option. This paper will help to test newer and updated applications of Lava's Android smart-phones. The main motive is to understand android smart phone and design a test framework according to the application. In Android each screen of an application is represented by an Android activity, so an Android application can be seen as an activity stack. Each activity itself consists of groups of ordered UI elements. Each activity has its independent lifecycle. As new features get added to mobile, testing becomes a challenge because of the complexity, speed to market and regression perspective. Hence the need for innovative means of testing is required.

Prior to customizing an android automated testing framework for lava's android smart-phones, first of all we need to clearly understand the android's default testing framework. The android automated testing framework is mainly controlled by the InstrumentationTestRunner [4] on the application package. At present android have different kinds of automated testing frameworks for testing different aspects of android applications. Some of them listed bellow-

JUnit, Robotium[6], MonkeyRunner [6], CTS etc. All of them use a basic framework of android system to tests the android applications. The Android testing framework is an integral part of the development environment provides architecture and powerful tools that help you test every aspect of your application at every level from unit to framework.

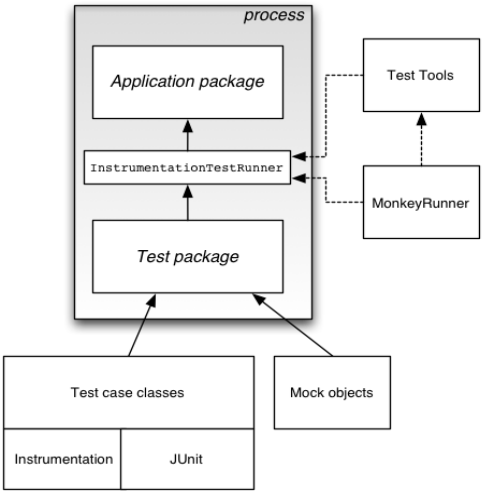


Fig 1: Android Testing Framework

The tests are conducted by the InstrumentationTestRunner, the test cases are encapsulated in a common test package and requests the InstrumentationTestRunner to perform those tests to the application package [5].

IJERT

II. IMPLEMENTATION

To implement this concept, we need to install the Android SDK, and Eclipse IDE also needed for both writing the test cases and to perform the various functionality of the android platform to tests the different features of the android applications of Lava's android smart-phones. The customization of automated testing framework is possible by the test driven development technique describes next with a flow diagram in this paper.

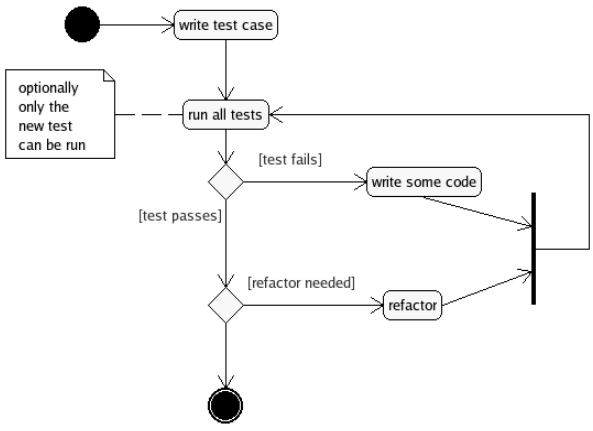


Fig 2: Test Driven Development

This concept has been developed in Java and is composed of three main components:

- A Java code instrumentation component

- the GUI Crawler
- The Test Case Generator.

The Java code instrumentation component is responsible for instrumenting the Java code automatically, in order to allow Java crashes to be detected at run-time.

The GUI crawler component is responsible for executing the Android crawling process. It produces a repository describing the obtained GUI Tree. Moreover, it produces a report of the experienced crashes, with the event sequences producing them. Robotium provides facilities for the analysis of the components of a running Android application. The GUI crawler extracts information about the running Activity, the Event Handlers that the Activity implements and the Widgets that it contains. Moreover, the GUI crawler is able to emulate the triggering of Events and to intercept application crashes.

The Test Case Generator component is responsible for the abstraction of executable test cases supporting crash testing and regression testing from the GUI Tree produced by the GUI Crawler component. Test cases produced by the Test Case Generator are Java test methods that are able to replay event sequences, to verify the presence of crashes (for crash testing) and to verify assertions regarding the equivalence between the Interfaces obtained during the replay and the original ones obtained in the exploration process (for regression testing).

III. RESULTS

The results of the tests are reflected by two ways either by the default JUnit interface or by generating a suitable test report. In JUnit the interface showing red signals for failure of test case and green for passing all the tests successfully.

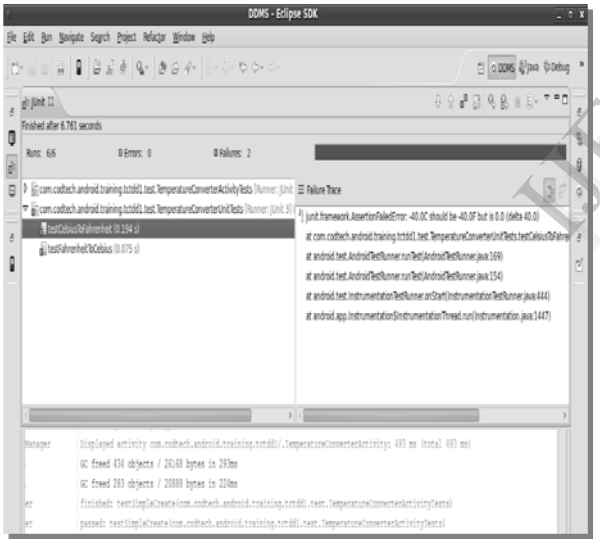


Fig 3: JUnit Test Report for Test Failure

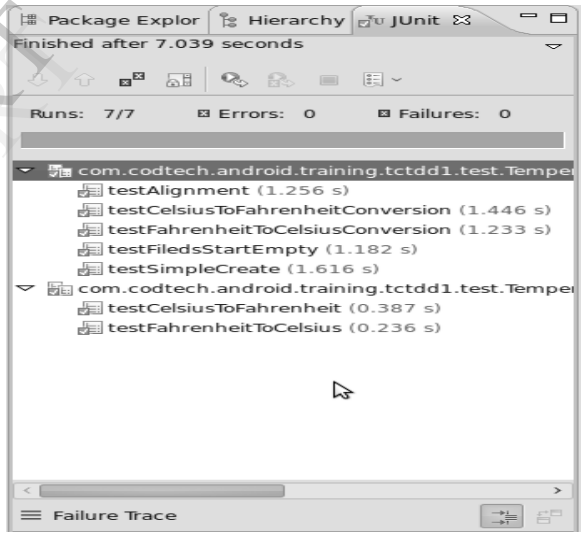
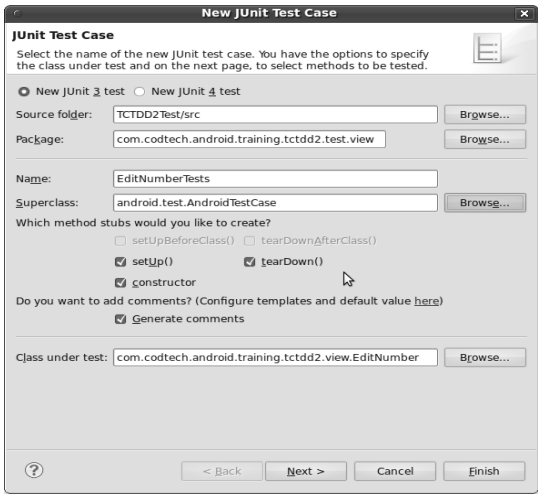


Fig 4: JUnit Test Report for Successful Test

- In the figure 3 an android application testing, the JUnit framework showing red signal as the result for a test performance, that means the test is failed with the android application, we can also see the details of the failure left side panel of the JUnit.
- In the figure 4 is showing green as the test over android application is succeeded.



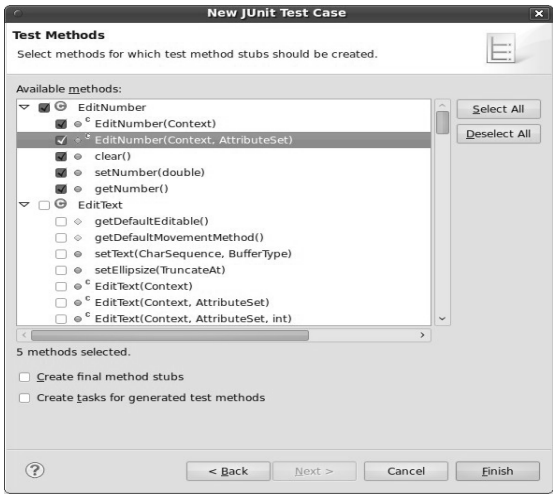


Fig 5: Writing a new JUnit test case

Fig 6: Selecting Test Methods

- The figure no 5 shows how to write the test cases for an android application using JUnit testing framework in Eclipse IDE. The "New JUnit 3 test" and "New JUnit 4 test" are the Junit framework version, always use New JUnit 3 test version because, it is stable "Name" is the name of the test project.

- The figure no 6 shows how to select the test method in JUnit testing framework to test the android application.

IV. CONCLUSION

- In this paper a technique for automatic testing of Android mobile applications has been proposed. Test cases consist of event sequences that can be fired on the application user interface. At the moment, we have not considered other types of events that may solicit a mobile application and just focused on user events produced through the GUI.

- The proposed testing technique aims at finding runtime crashes or user-visible faults on modified versions of the application. In order to detect runtime crashes. Moreover, in order to increase the effectiveness of the obtained test suites we intend to investigate further and more accurate techniques for the crawler to generate several kinds of input values, including both random and specific input values depending on the considered type of widget. In addition, solutions for managing test case preconditions and post-conditions related to persistent data sources will be looked for.

V. REFERENCES

[1] Domenico Amalfitano, Anna Rita Fasolino, Porfirio Tramontana A GUI Crawing based technique for android Mobile Application Testing. ICST - International Conference on Software Testing, Verification, and Validation 2011 2011 IEEE Fourth International Conference
[2] Google Inc. What is Android? http://developer.android.com/guide/basics/what-isandroid.html, Nov 15, 2011
[3] Automated GUI Testing on the Android Platform. Author(s): Kropp Martin | Morales Pamela Journal: IMVS Fokus Report ISSN 162-2014. Volume: 4; Issue: 1
[4] Charlie Collins, Michael Galpin, Matthias Kappler, "Android In Practice", Manning Publications Co., 2011, PP 601-610.
[5] Diego Torres Milano, "Android Application testing guide", PACKT publishing Ltd., June 2011, PP 27 – 47.
[6] Google Code. Robotium framework. About Robotium, Accessed May 2012, http://code.google.com/p/robotium/
[7] Android Source Code, March 2012, http://source.android.com/.
[8] Highsoft Solutions AS, Licensed for free under Creative Commons Attribution Non Commercial 3.0 License, http://www.highcharts.com/license, November 25, 2011.
[9] Open handset alliance, accessed May 2012, http://openhandsetalliance.com/.