Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

EMPIRICAL STUDY ON AUTOMATED GUI TESTING TECHNIQUES FOR ANDROID

Madiha Yousaf [a], Muhammad Haris [b]
[ab] COMSATS Institute of Information Technology, Islamabad, Pakistan
Corresponding email: m.haris@comsats.edu.pk

Abstract

Smart phones with high-resolution touch screens have become a vital part of everyone's life. Android technology in smart phones provides advantage of being open source and user friendly. Android works with hundred of devices where users can test and modify their phone as they need. The important aspect of a mobile phone is not only how it meets the user requirement, but the most important is the correctness and accuracy of its response and interaction with graphical user interface. The traditional GUI testing of today's smart phones is not enough to test the Android application. Android application is actually event-driven; so it is necessary to assess how techniques can be adopted to carry out cost-effective testing processes in the Android platform. There are special automated techniques for GUI testing of android applications proposed by software engineers. This paper presents an overview of recent work done in the area of automated GUI based testing of Android applications.

Keywords: Android, GUI Testing & Automated Testing Techniques.

1. Introduction

The worldwide market of Smartphone shows that Android is the most popular platform. According to International Data Corporation in the second quarter of 2016 number of smartphones shipped account to 343.3 million in which market share of Android OS accounted for 87.6% (IDC.Com, 2016). Android devices like e-readers and tablets are also going popular rapidly. Android devices are becoming part of everyday life for different types of use such as information sharing, casual gaming, online transactions, audio and video playback and other monetary transactions. The growth of the Android Market exceeds 1 billion app downloads per month (Chu, 2011). The increase in this trend is basically due to the user friendly, appealing, colourful graphical user interface provided by these mobiles apps. Thus it is very important to make sure that the graphical user interface is reliable and the applications are running on it smoothly. The question that now arises is how to ensure graphical user interface reliability of android mobiles!

Manual testing let the user perform random testing but it is error prone and time consuming. Manual testing is resource consuming including a lot of human effort, time, trained teams, and proper expert team-lead to get best results along with sound knowledge of development and users experience.

There are many manual testing techniques but the issue arises due to the non-deterministic nature of the android as it becomes impossible to test manually for every possible event as it will be quite lengthy and time consuming. It is not really possible for a human to test all possible inputs criteria and to retest all of them again after the bug has been fixed. So, there is a requirement for automated testing for graphical user interface to ensure the reliability and correct behaviour of the GUI's, as graphical user interface have become an important factor for success or failure of the android phone.

 DOI: 10.25275/apjcectv3i2ict1

Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

The Graphical User Interface (GUI) of the Android app is an essential component for automated testing techniques. GUI is the screen viewable to the user and it provides all or many use cases of the application to the user. Most of the mobile applications nowadays are GUI driven or GUI centric. They are not designed to be exercised without GUI. Considering the GUI alone i.e. without analysing applications implementation is sufficient to extract the use cases. As the GUI of mobile apps is complex in functionality, manual testing requires intensive resources. Hence automated GUI testing techniques are required for the better quality of Android Apps (Yumei & Liu, 2012).

Automatic testing of GUI needs to mimic human interaction with the GUI. It is done by using widgets and verifying the GUI. GUI can be verified by using an application programming interface, optical character recognition or bitmap comparison. There are different tools for testing GUI present in the android software development kit for example monkey testing tool, Hierarchy viewer etc, but they have some limitation for system level testing.

This paper presents us with the review on automated GUI testing techniques of Android Applications. Section 2 provides us with the different kinds of automated testing techniques available for GUI testing as well as a little classification showing different types of techniques used for different purposes. Section 3 of this paper provides us with the discussion and a tabular view of different techniques and in which scenarios they can be used. Section 4 provides us with the conclusion and future work.

2. Automated testing techniques for GUI testing in Android

Many testing techniques have been implemented and presented for automating the GUI testing for android. Although techniques like android verification, GUITAR, GUI testing and android bug studies are already there, these are still lagging behind as these studies are looking for where the bug is present not looking for semantics and their fixation. So here we have studied many different approaches and techniques for automated GUI testing for android which will be discussed here as well as we have also studied the effectiveness of the evaluation of the testing. Before explaining these techniques let us provide here a classification of different techniques based on their types. We have differentiated various techniques on the basis that either they are taking test cases or are based on some tools or models. The table below gives us an overview of the technique and its type.

Table 1: Testing Techniques

Technique Name	Technique Type
Bug study and bug detection	Test case and event generation
Android Instrumentation framework	Test case and event generation
Positron Framework	Client server based model
Testing based on Image flow	Images are taken as input
Silkuli	Tool Based
TEMA Tools	Model based
Model based GUI testing for web based android applications	Model based
Android Ripper	Tool based
Grey-box Approach	Model based
GUI Tool Set	Model based
ART test case generation technique	Tool based technique

 DOI: 10.25275/apjcectv3i2ict1

Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

GUI crawling based technique	Model based technique for android application
Accessibility-based approach	Tool based approach

3. Bug study and bug detection

A technique already presented for testing GUI for android phones becomes a new idea as in this study first the bug collection and categorization has been carried out based on ten popular android applications. After the bug study has been conducted the bugs are then categorized into various categories out of which the GUI bugs are considered for further bug detection using test case and event generation. GUI bugs are identified as activity, event and type error bugs. The dynamic approach presented consists of the many techniques. The first step includes the automated test case generation which is done by using JUnit. Once the test cases are there then the events are generated as for GUI detection events plays a very important role, thus for this purpose monkey event generator has been used. After this the test cases are run on the Dalvik virtual machine and the traces are generated which are then saved in the log files. Then comes the log file analysis which determines the bugs based on the specified patterns. The biggest advantage of this approach is that in addition to the old bugs, new bugs are also identified in the less time (Hu & Neamtiu, 2011).

4. Android Instrumentation Framework

The Android Instrumentation Framework is an incorporated fragment of the android software development kit. Instrumentation tells us about the competence of monitoring and diagnosing an application by giving as input the debugging techniques, tracking code and performance counters which helps us in measuring the performance and in controlling the application's behaviour. It makes use of the JUnit assertions to validate the GUI behaviour and state. No new verification models are introduced, making it simpler to be used for expert JUnit tester.

4.1 Positron Framework

Coming to the positron framework it is a client-server model made above the Android Instrumentation Framework to pay attention to the activity's sources, giving Selenium an improved approach for writing and running test cases. Apiece test case is served as client, which associates with the server that runs that particular activity. In addition it also facilitates us with the communication infrastructure and server services, which allows us to carry out the tests self-reliantly of one another. The major contribution of this approach is that they have provided us with the detailed comparison of different approaches and made it quite easier to know at which stage we have to use which framework for testing. The two explored frameworks make available basic GUI testing functionality on various levels. Compared to GUI desktop testing tools both frameworks still show notable limitations (Kropp & Morales, n.d.).
The Table 2 sums up the comparison of the two approaches.

Table 2: Comparison of instrumentation and positron framework

Android Instrumentation Framework	**Positron Framework**
Low-level API to simulate user interactions with the screen	Abstracted comfortable high-level interface for writing GUI tests
Direct handle to the context used to poke freely around the activity to validate test assertions	Connects to the application under test each time when the test class needs to use activity resources which slows down test runtime execution.
Every UI element must be brought	Locate activity resources by calling getters for each named

 DOI: 10.25275/apjcectv3i2ict1

Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

individually in the setup method	property class
Offers greatest flexibility and direct access to the GUI controls	Reduces the effort, for both, writing and maintaining test code significantly

5. Image based testing

Another approach on which work has been done previously includes the concept of automating GUI testing for Android applications based on the image flow. The above explained concepts are related with the test cases and client server models, here we are using the concept of images. In this approach the images on the mobile are encoded to be transferred to the server and then data in the byte pattern is changed to the pixels and then the storage in the XML files have finished. Developer detects the created Image Flow and then differentiates the existence and nonexistence of errors. XML is being used for the Image Flow made previously in the following testing. As the second testing is being carried out by relating XML for Image Flow prepared by this time and novel XML, extra and eradicated parts and errors of images are eminent. Thus this approach can be used for further GUI testing in mobiles and for integrating it in different environments and machines (Kwon & Hwang, 2008).

5.1 Evaluation of automated testing

In addition to the above explained automated testing techniques for android GUI's another approach is there which makes us familiar with the effectiveness of making the testing for GUIs automated. Effectiveness is found by keeping different requirements in the mind which includes capable to run tests on two phones in place of one, carry tests on dissimilar products of the similar product acceptable to measure the reusability of test models, comprise test data in test run in an simple and easy way and attempt to originate test model commencing design models. Results show that more than half of the bugs were revealed in modelling and the remaining third by test execution (Jääskeläinen, et al., 2009).

6.Sikuli

Sikuli is an open source automated GUI testing technique. It gives us an environment to write and execute a script using Jython code, utilizing screenshot pictures (.png) as a function inputs. Jython uses the Python programming language and allows it to execute on the Java platform. Sikuli IDE gives an interface, to create, execute scripts in step-by step manner. Two main functions performed by sikuli script are the keyboard or mouse actions performed by Java Robot class, and to find images with C++ pattern matching algorithms using OpenCV.

Sikuli in comparison with other GUI testing technique e.g AutoIt, Whyline, CoScripter, Guitar, Robotism, NativeDrive, doesn't need to know the source code, name and stored position of the application and the system calls it invokes. The EBL (embedded benchmarking lab) project successfully introduces a refined way of android GUI testing using Sikuli technique by eliminating two of its basic limitations of inability to record the test cases performed by user, and lack of output result analysis.

Sikuli is a powerful tool, it uses simple syntax of Python. Sikuli uses the full cross-platform functionality of Java. It also adds its own libraries. It is easy to use on any platform/operating systems. Sikuli script allows black box testing. We don't need to know anything about the program but its appearance and visual behaviour. Being an open-source tool, it has a great chance to develop rapidly. Sikuli library gives a great foundation for future development. Making new testing tools, as well as improving the existing ones, has a great potential for researchers and developers. The disadvantage of Sikuli is that most error occurrences are due to failure of image recognition. The current stage of image recognition technology is still not efficient to be used for wide usage. In an android testing it was found

 DOI: 10.25275/apjcectv3i2ict1

Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

that the number of error occurrence reached up to 50%. Out of which 4/5 were due to image mismatch. Even though the rate is higher than for Windows system, it is still quite good considering that there aren't many testing tools for Android, while the existing ones have very limited capabilities (Volodymyr & Ying-Dar, n.d.).

7. TEMA Tools

TEMA web GUI is used as an interface for testers having a test server, which is used for managing and designing test patterns, more over executing and following the actual tests, and test model packages are managed. This all simmers down letting testers immediately make the choice of what physical device they want to execute their tests on and what they would like to test.

TEMA tool is used for model based testing of GUI. It has four phases of model based test i.e. modelling, design, generation and debugging. The phase for test modelling have tool for designing and utilities of model. Second phase of design contains a web GUI and it designs the objective of tests. The test creation phase deals with a number of techniques and algorithms that uses model to generate desired tests. The debugging phase interprets unsuccessful test runs. The models designed in TEMA are represented by state machines, showing transitions between different states.

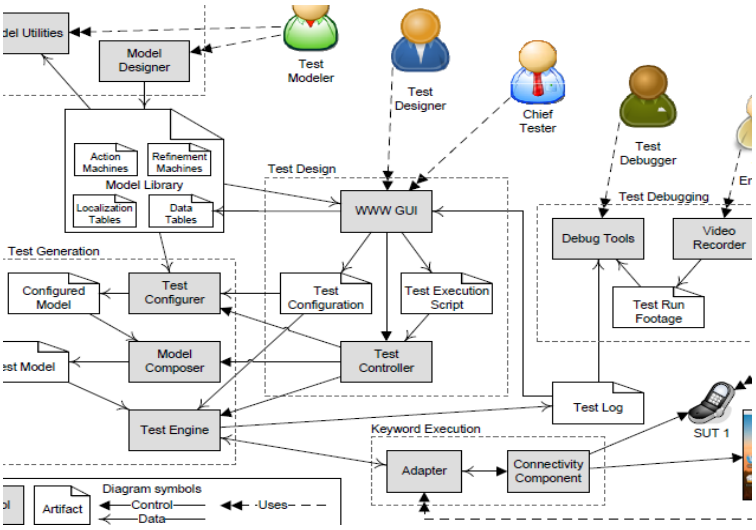

Figure 2: Architecture of TEMA TOOL

Action words describe the action/interaction of user with the GUI at a high level of generalization, for example sending an SMS, making call etc. Action words are then converted into a series of keywords for menu direction-finding, text input etc. Keywords are used in low level models. TEMA uses action words and keywords in models at diverse levels of abstraction. Action machines that contain action words are formed with modification machines having key words. The resultant composite model is taken as an input to the tools executing the model. This has been implemented by making use of an on the fly algorithm to avoid state space explosion. TEMA approach is also suited to the domains other than mobiles. TEMA has been tested on BBC News Widget; an android application. 14 bugs were found in the application, 6 during test execution phase and 8 during modelling process. The model precisely found even small discrepancies, which might have been neglected by manual testers.

 DOI: 10.25275/apjcectv3i2ict1

Model library of TEMA contains models for over 100 action machines containing almost 1000 action words. Action machines can be used again for different domains. TEMA testing covers more bugs at modelling level that at execution level. It gives reduced amount of test maintenance (Katara, 2011).

8. Model based GUI testing for web based android applications

A German company Heidelberg Mobile International has been working in mobile industry for over more than 12 years. The company has developed an online information system for pedestrians. It's a mobile location based, user-friendly, cross platform web based information system application. UI testing of this application has been done manually by quality assurance team.

The research question of the paper is that does really the automated model based GUI testing give efficient results and what will be the behaviour of application under test i.e. CeBIT2go mobile web application. Model based testing test a system in abstraction level. It uses model based design particular for an application to perform testing. Model is an abstract of the software under testing SUT. The most important factor for android automated testing is to properly understand the system under test, therefore model based testing is best suited for GUI testing of android applications as it works on a separate model designed for each application/system.

CeBIT2go is a mobile web application in which with the mobile ticketing enabled the user enters into an exhibition, the application tracks user location, and also the mobile payment is activated when user purchases something from exhibitor. The interface enables multi-touch by providing some buttons, image icons, list and a map that could freely be moved around with figure. Selenium is used for providing automatic web based administration tasks. The application under test was examined under three main tests i.e. verify the elements of page such as button icons after page loading, page transversal, and functionality testing for specific functions. For the functionality testing 17 test cases were made to test application on Samsung galaxy Tab manually and automated. Results showed that automated testing spent more time to the test cases which deals with page loading, UI element location or element assertions. In manual testing once the page has been loaded testers can only scan through the application page. Automated testing takes much less time as compared to the manual testing for test cases involving web browsers or zoom in and out functions. Therefore automated testing gives better results with web dependent mobile applications as the web interface and GUI for android application have similar structures. In automated testing, test cases take the least amount of time to execute, thus making implementation easy. It also provides benefits of reusability and maintainability (Methong, 2012).

Android RipperAndroidRipper [11] is an automated technique implemented in a tool that tests Android apps via their Graphical User Interface (GUI). It uses ripping for the purpose of automatically and systematically traversing the app's user interface. And then generate and execute test cases as new events occur. The aim of the AndroidRipper is to obtain the sequence of events that are generated through GUI widgets. This has been done by analyzing the application's GUI dynamically. Each generated sequence provides an executable test case. During the ripping process, AndroidRipper generates a GUI Tree model by maintaining a state machine model of the GUI. This GUI tree model consists of GUI states and state transitions occurred during the process. The iterative process of GUI Ripping is as follows:

 DOI: 10.25275/apjcectv3i2ict1

Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

```
Task List Initialization;
while (Task List Is Not Empty) {
    Extract a Task From The Task List;
    Execute the Task;
    Abstract the Current GUI Abstract State;
    Update the GUI Tree Model;
    if (GUI Exploration Criterion) then {

Define New Tasks;
Add New Tasks To The Task List; } }
```

This technique provides good level of code coverage i.e. around 37% to 39% and real bugs and crashes detection. Its testing is compatible with smoke testing process and is more effective with respect to standard random testing tools such as Monkey.

9.Grey-box Approach

Another approach for automated testing of Android app is the Grey-box approach of GUI model generation [12]. It uses the static analysis to extract the set of events carried by the GUI of application and by the of technique dynamic crawling to generate the model of application by methodically executing the extorted events on the running application. In Android framework, user actions are defined by recording a suitable event-listener for it or by taking over event handling technique of an Android framework component. Identification of both type of actions are performed by identifying place where act is initiated or registered, finding the component on which the action execution has been done and the identifier being extracted. The output of this static analysis is action map which contains the IDs of components on which the action are executed. The GUI model is developed as finite state machine of the behaviour of Android app. The crawling algorithm discovers all the app's states by firing open actions on every pragmatic state. The crawling process comes to an end if there are no open states in the model which means there is no state which have open actions to be executed. The output of the algorithm is a crawled model of the app. It takes 70% less time to traverse the code then DFS and increase the coverage by 34% on average. This technique provides the reusable quality model for Android app testing but it requires a one time manual effort to select the attributes of executable components to compose the visual observable state of the GUI.

10.GUI Tool Set

The authors of AndroidRipper improved their technique and presented a GUI toolset for testing Android applications [13]. They worked on extending the automation level of previous technique and developed a tool set in which the manual intervention of the tester is quiet reduced. A modular GUI ripper is designed for that purpose, consists of nine components, giving GUI tress and crash report as final output. An automated test method is developed to use GUI Ripper in testing Android apps with the help of tool set. The first step is deploying, in which a testing program is obtained that is executable on Android Virtual Device (AVD). A set of tools are employed in this step that includes Source Code Instrumentation tool, ripper options configuration tool and deployed tool. Second step is ripping, which executes the testing program on AVD from the initial state and provide source code coverage and crash reports. Ripper Executor Tool performs this automatically. The last step is post-processing, this provides report of code coverage, GUI Model and JUnit test suite. The tools involved in this step are code coverage generator, GUI Model translator and JUnit test Suite Generator. This technique gives 53 to 69% of code coverage and detected new bugs as well.

 DOI: 10.25275/apjcectv3i2ict1

Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

11. ART test case generation technique

The researchers proposed a model using black box testing for mobile application and further presented an automatic test case generation technique by prolonged the adaptive random testing. For generating test cases automatically, distance metric and a test case generation technique that is named ART is introduced to that. ART algorithm is an algorithm adopted from the Chen et al. algorithm (FSCS-ART) and basically depends on distance of test cases and information of input. They applied adaptive random testing to model the inputs of mobile application and generating the test cases. They have mainly concerned with the mobile application inputs either GUI event or context event. ART algorithm for generation of test cases is implemented by smart-monkey tool in Mobile Test. ART technique for test case generation and event sequence distance both are not only appropriate for mobile applications testing, but also appropriate for the embedded event-driven software. It reduced the number of test cases (Domenico, et al, 2012).

12. GUI crawling based technique

An automatic testing technique is proposed that based on graphical user interface (GUI) crawler. In user interface the GUI crawler based technique pretends events of real user and automatically assumes a GUI model. The GUI model is executed automatically and generates test cases that may be used in regression and crash testing. A GUI tree is generated by GUI model that starts deriving test cases for crash testing as well as regression testing. The GUI tree nodes represent the user interfaces and edges represent the transition between events of Android application. Android Automatic Testing Tool (A2T2) is also introduced that supports their proposed technique. Three main components of A2T2 are GUI crawler, instrumentation component of java code and Test Case Maker. The A2T2 prototype accomplishes Widgets subsets of an Android application. The advantage of the above technique is that it detects runtime crashes faults and also shows effectiveness in automatically detecting the faults. But in real applications the implementation of crawler based technique is required by experiments that show the technique scalability. (Amalfitano, et al., 2011).

13. Accessibility-based approach

An approach is proposed for testing tools of GUI that used a visualization mechanism and for retrieving and monitoring GAPs used accessibility technologies. Accessibility technologies are used because they provide objects of GUI and generate the events and also set the values. The basic aim of this approach is that, just like human-driven procedure, it makes it easier to create testing tools of GUI for engineers. Users read or enter data into GUI objects when an initial screen appears. By causing some actions the users initiate transitions. Microsoft Active Accessibility (MSAA) is accessibility technology that is used in their approach for windows. By Accessibility technologies SMART (System for Application Reference Testing) and REST (Reducing Effort in Script-based Testing) tools are created for GUI testing. Smart specified that how user use GAPs reference and how they used it for other input data. REST is used for testing Gaps modification in test scripts (Grechanik, et al., n.d.).

14. Discussion

Different successful techniques for automated GUI testing of android application are reviewed. Concluding our findings we can say that manual testing has been left far behind and the automated testing for GUI's have made it quite popular due to its distinct features as well as due to the various techniques available for different kind of data present in our GUI. Important aspects of some techniques are summed up in the Table 3. This summarized table will help the android testers to get information about the latest different automated GUI testing techniques, their effectiveness and performance.

 DOI: 10.25275/apjcectv3i2ict1

Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2017 , Volume: 3, Issue: 2

Table 3: Different Automated GUI testing techniques for Android

Technique name	**Technique Type**	**Novel Idea/ Modification**	**Tested on**	**Performance Results**
Bug study and bug detection	Test case and event generation	Novel idea	Android framework using JUint and Monkey event generator	Effective as after automating new bugs are detected in addition to old one
Android Instrumentation framework	Test case and event generation	Novel idea	Android framework using JUnit assertion functionality	Effective in low level API and provides greatest flexibility and direct GUI controls
Positron Framework	Client server based model	Modification of android instrumentation framework and uses selenium like commands	Android instrumentation framework	Effective in abstracted high level interface and reduces the effort for both writing and maintaining.
Testing based on Image flow	Images are taken as input	New idea	Mobile framework by making tool having image flow server and encoder	As by comparing the new image with the already stored ones errors can be found out successfully.
Silkuli	Tool Based	New research project, of User Interface Design Group, MIT Computer Science and (CSAIL), National Science Foundation.	EBL (embedded benchmarking lab) project	Sikuli is a powerful tool it uses simple syntax of Python. Sikuli uses the full cross-platform functionality of Java. Sikuli in comparison with other tool based GUI testing technique doesn't need to know the source code, name and stored position of the application and the system calls it invokes.
TEMA Tools	Model based	Novel idea	BBC News Widget	TEMA testing covers more bugs at modeling level that at execution level. It gives reduced amount of test maintenance. The model correctly found even minute

 DOI: 10.25275/apjcectv3i2ict1

				discrepancies, which might have been neglected by manual testers.
Model based GUI testing for web based android applications	Model based	Novel idea	CeBIT2go mobile web application	Model based automated testing gives better results with web dependent mobile applications as the web interface and GUI for android application have similar structures. In automated testing test cases take least time to execute thus making implementation easy. It also provides benefits of reusability and maintainability.
AndroidRippe r	Tool based	Extended GUI Ripping [7]	Android App “Wordpress”	Better coverage of LOC i.e. around 37 to 39% within almost same time duration as in Monkey testing.
Grey-box Approach	Model based	Novel idea	Android Apps TippyTipper, OpenManager, Notepad, TomDroid, AardDict, HelloAUT, ContactManager, ToDoManager	Take 70% less time to traverse the code then DFS, increase the coverage by 34% on average
GUI Tool Set	Model based	Improved version of AndroidRipper	Android Apps AardDict, TomDroid	Increased coverage of LOC i.e. around 53 to 69% , new bugs found.
ART test case generation technique	Tool based technique	Adaptive Random Testing	Implemented as a tool named smart-monkey within MobileTest	ART use smaller amount of time than random to depict the first fault across all application and reduced test cases
GUI crawling based technique	Model based technique for android application	New technique by adapting existing GUI techniques	Through a tool A2T2 tested a small size Android application.	Usability for running crash testing and failure testing, efficiency in identifying some types of errors in a completely automatic

 DOI: 10.25275/apjcectv3i2ict1

				manner.
Accessibility-based approach	Tool based approach	Novel idea	System for Application Reference Testing (Smart) & Reducing Effort in Script-based Testing (REST)	To help test engineers it issues a warning to fix errors in test scripts.

Conclusion

Android technology is getting advanced rapidly thus it has become the fundamental requirement of one's life. Android applications provides user friendly and effective GUI to users which is the key factor of its success. Assuring the reliability of GUI for android has become the most important and crucial task for android developers. Automated testing is the most advanced and adapted technique for GUI testing nowadays. This paper presents an overview of different automated GUI based testing techniques for android applications that could be helpful for android quality assurance team.

 DOI: 10.25275/apjcectv3i2ict1

References

i. Amalfitano, D., Fasolino, A. R. & Tramontana, P., 2011. *A GUI Crawling-based technique for Android Mobile Application Testing. IEEE Fourth International Conference*. s.l., IEEE.

ii. Chu, E., 2011. *10 Billion Android Market Downloads and Counting*. [Online] Available at: http://androiddevelopers androiddevelopers.blogspot.com/2011/12/10-billion-androidmarket-downloads-and.html [Accessed 6 November 2016].

iii. Domenico, et al, 2012. *A toolset for GUI testing of Android applications. Software Maintenance (ICSM), 2012 28th IEEE International Conference on. IEEE*. s.l., IEEE.

iv. Domenico, A., 2012. *Using GUI ripping for automated testing of Android applications. Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. s.l., IEEE/ACM.

v. Grechanik, M., Xie, Q. & Fu, C., n.d. *Creating GUI Testing Tools Using Accessibility Technologies. IEEE International Conference on Software Testing Verification and Validation Workshops*. s.l., IEEE.

vi. Hu, C. & Neamtiu, I., 2011. *Automating GUI Testing for Android Applications*. Waikiki: Honolulu.

vii. IDC.Com, 2016. *Android dominated the market with an 87.6% share in 2016Q2016*. [Online] Available at: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

viii. Jääskeläinen, A. et al., 2009. *Automatic GUI Test Generation for Smartphone Applications – an Evaluation*. s.l., IEEE.

ix. Katara, T. T. a. M., 2011. Experiences of System-Level Model-Based GUI Testing of an Android Application. *IEEE*, pp. 377-386.

x. Kropp, M. & Morales, P., n.d. *Automated GUI Testing on the Android Platform. Institute of Mobile and Distributed Systems*. Northwestern Switzerland: University of Applied Sciences.

xi. Kwon, O. & Hwang, S., 2008. *Mobile GUI Testing Tool Basedon Image Flow. Seventh IEEE/ACIS International Conference on Computer and Information Science*. s.l., IEEE.

xii. Liu, Z., Gao, X. & Long, X., 2010. *Adaptive Random Testing of Mobile Application. Computer Engineering and Technology (ICCET) 2nd International Conference*. s.l., ICCET.

xiii. Methong, 2012. Model-based Automated GUI Testing for Android Web Application Frameworks. *International Conference on Biotechnology and Environment Management,* Volume 42, pp. 106-110.

xiv. Volodymyr, A. & Ying-Dar, L., n.d. *Automatic Functionality and Stability Testing Through GUI of Handheld Devices*. Chiao Tung, National Chiao Tung University.

xv. Yang, Wei, Prasad, M. & Xie, T., 2013. *A grey-box approach for automated GUI-model generation of mobile applications. Fundamental Approaches to Software Engineering*. s.l., Springer Berline Heidelberg.

xvi. Yumei, W. & Liu, Z., 2012. *A Model Based Testing Approach for Mobile Device. Industrial Control and Electronics Engineering (ICICEE)*. s.l., IEEE.