Effective Mobile Applications Testing Strategies

Haeng-Kon Kim[1], Hyun Yeo[2], Tai-Hoon Kim[3], Ha Jin Hwang[4], Carlos Ramos[5] and Goreti Marreiros[6]

[1]Department of Computer Engineering, Catholic University of Daegu, Korea
[2]Department of Computer and Communications, Sun Cheon National University, Korea
[3]Department of Convergence Security, SungShin Women's University, Seoul, Korea
[4]Department of Business Analytics, Sunway University Business School, Malaysia
[5-6]Institute of Engineering, Polytechnic of Porto, Porto, Portugal
[1]hangkon@cu.ac.kr, [2]yhyun@sunchon.ac.kr, [3]taihoonn@daum.net ,
[4]hjhwang@sunway.edu.my, [5]csr@sc.ipp.pt, [6]mgt@isep.ipp.pt

Abstract. In highly fragmented and competitive global market the mobile development cycle is of short period. For the vendor's equanimity and overwhelming task to ensure long term success, the APP must be tested over different combination of platforms, operating systems and networks before launching to global. In addition to this, similar to Functional testing the non-functional testing like Security testing, usability testing etc. also plays an important role. The effective test planning in Mobile Application testing makes helps to improve the quality of Mobile Apps. Due to increasing numbers of smart phone applications and their advanced features, smart phone has become a primary resource of communication for worldwide business owners, industrialist, office workers, etc. In this paper we use UML diagrams represented in the form of a tree to extract test cases to verify/validate the behavior of mobile objects concerned. A new model based approach for automated generation of test cases in object oriented systems has been presented. The test cases are derived by analyzing the dynamic behavior of the objects due to internal and external stimuli.

Keywords: TDD(Test Driven Development), UML Test Case, Mobile Applications Testing, Genetic Algorithm, Functional/Non Functional Testings

1 Introduction

Mobile applications code-based test generation has made tremendous progress in the recent past: Today, modern systems are able to generate inputs that drive execution to almost any point in the control flow. Automation, however, reaches its limits when it comes to controlling the test's environment: For example, if the tested code accesses an external component such as a service or a database, then this component needs to be controlled by the test as well in order to prevent unwanted side-effects like data loss. A common solution in test generation is to create a stub version of the component that is difficult to control. Such a stub object provides the same interface as the component it represents, but returns predefined values on method calls. In this paper, we propose an approach that automatically generates such stub objects, helping to drive test generation towards its goal in cases where automatic generation is difficult or impossible otherwise. The solution applied is to map the stubbed behavior to the input space of the test generation problem: For each method call on the stub object in the test case we need to find an appropriate value that the stub returns during execution. A main difficulty in this approach is that within the scope of a test case the stubbed component should behave similarly to the real component.

2 Related Works

2.1 Key Mobile Testing Challenges in Mobile App Test Automation [1]

The primary factor that determines an automation tool's success is its ability to work across platforms and technology stacks. The following challenges influence automation success:

1) Planning of Quick Rollouts:
The companies are looking for the golden business opportunities in unique Mobile apps and expecting rapid rollout of quality application or improvements and bug fixes if application is already launched. They want to push the applications in the market as quickly as possible to avail the benefits of market boom mobile sector. As a result the QA testing cycle which generally takes two to three weeks depends on the complexity and the size of the application is now reduce to half or one week. Due to clutch in the timelines the QA it is very difficult to problems if mobile applications don't meet the customer expectations.

2) Multi-Platform Compatibility:
With the propagation of mobile devices like iPhone, iPad, Smartphones, Tablets, Windows Mobile and wide range of Andriod devices etc, mobile application providers have to provide the multi platform compatibility to reach their audience. In the mobile industries there are no any industry standards for Operating Systems or device hardware, so testing of apps over variety of devices is not a simple task. So here we cannot 100 % say that test cases which passed for one device are also passing for other devices, even if device from same family.

There are many combinations while testing mobile apps like Screen resolution, memory sizes, battery, Operating System etc. The creation of separate test case and execution on each device can be the most expensive and time consuming task.

The mobile application market is rapidly growing with a demand of quality product with no any excuses on errors and security holes.

3) Dealing with a variety of connectivity modes:
One more important parameter to be considered in the mobile testing is the "Modes of Connection" to access the application. This step can be ignored if the internet connection does not required for application under test, however almost all applications requires internet so this test case needs to run over different connections

like WiFi, 3G, 4G etc. Even you test the application you will face the wide range of applications over different connectivity options. While planning you QA Automation testing strategy you need to consider connectivity modes which are equally important.

4) Creating end-to-end tests:

The mobile market demand is to integrate the mobile applications with all platforms and expected to flawlessly access the data on mobile and other platform like Web site. The end to end test cases should be work as expected on mobiles. Consider a example where order is placed from the mobile device and same can be from the log in into Web site. These mobile apps are expected to work on front end and back-end systems.

2.2 Types of Mobile App Testing [1]

1) Functional Testing:

Functional testing performs on the functional behavior of the application to ensures that the application is working as per the requirements. Mostly, testing performs on the user interface and call flows of the application. As like other UI applications mobile applications also require lots of human consideration. If, functional testing performs on mobile devices manually, not automatically, it is going to be extremely complex, exhaustive and time-consuming task due to various mobile-specific challenges like; various mobile devices, mobile operating systems, and functions & applications involve with mobile devices. Functional testing automation process also requires lots of human resources, money, and time then too testers are ready to automate the testing process by using many tools due to its strong market value and user demand. Teams can then combine automated tests with selected manual test scenarios to balance the coverage and efficiency of the functional testing. To test some functionalities of the application tester go for manual testing process, later on tester combines manual testing and automation testing for better result.

2) Performance Testing:

The testing process is carried out by tester to test the performance and actions of the applications that pass through various mobile device challenges like; low battery power due to heavy battery uses, network out of coverage area/poor bandwidth/changing internet connection mode (2G, 3G, or WiFi)/changing broadband connection, transferring heavy file, less memory, concurrent approach to the application’s server by various users, etc. Application’s server and client both strongly affect the performance of the mobile application, so testers perform testing on both side of the application.

3) Usability testing:

Usability testing is used to test the mobile applications in terms of usability, flexibility, and friendliness. The testing process makes sure that the mobile app is now easy to use and offers a suitable user experience to the customers.

4) Installation testing:

Mobile devices hold two types of applications; the one which automatically comes with mobile OS (while installing OS, it automatically get installed), and another one you have to install specially from the store to use the particular application.

Installation testing is used to test the particular application is installing, uninstalling, and updating properly without any interruption (user is smoothly and flexibly installing the application).

5) Operational Testing:

Any mobile OS and desktop OS provides in-built back-up and recovery operational functions that save or recover all files or doc of mobile devices or applications that had been lost due to some reason. Operational testing is used to test that the particular back-up and recovery process is working properly and responding as per the requirement

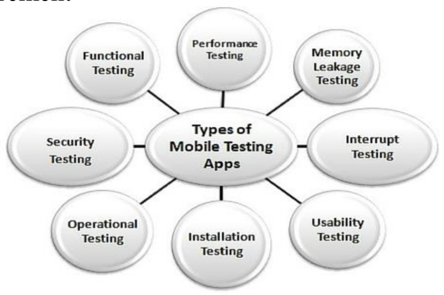

Fig. 1. Types of Mobile App Testing

3 Mobile Applications Automated Stubbing

3.1 UML Test Case Generation

In software development, use cases define system software requirements. Use case development begins early on, so real use cases for key product functionality are available fairly early in the project. A use case fully describes a sequence of actions performed by a system to provide an observable result of value to a person or another system. Use cases tell the customer what to expect, the developer what to code, the technical writer what to document, and the tester what to test.

For software testing, creation of test cases is the first step. Then test scripts (collections of test cases) are designed for these test cases, and finally, a test suite/plan is created to implement everything.

Test case (TC) A set of test inputs, executions, and expected results developed for a particular objective.

Test Script/Procedure. A document, providing detailed instructions for the [manual] execution of one or more test cases.

Test suite. A collection of test scripts or test cases that is used for validating bug fixes (or finding new bugs) within a logical or physical area of a product

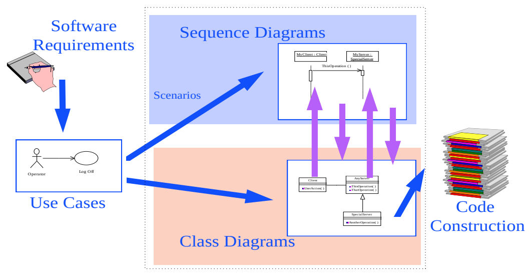

Fig. 2. UML Test Case Generation

4 Conclusion

All above testing methodologies prove that users can trust on applications that come with mobile devices, all applications are completely tested with many testing methodologies. But, be careful before using applications on mobile devices, if mobile applications involve internet connection then make sure that the device is already carrying antivirus. If, you don't have antivirus on your device then it won't be a fault of the applications installed on the mobile device; it is just a fault on you mobile system.Mobile automation testing is not so easy task; it requires proper planning, research, and practices to make the testing successful. Before to start the testing process; tester should be having good knowledge of information technology skills, the application on which you are going to perform the testing operation, and the tool that you are going to use to test the application.

<Acknowledgement>

This research was Supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the C-ITRC (Convergence Information Technology Research Center) support program (IITP-2016-H8601-16-1007) supervised by the IITP (Institute for Information & communication Technology Promotion).

This research was also supported by the International Research & Development Pro-gram of the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (Grant number: K 2014075112).

This work was also supported Europe-Korea Research on Ubiquitous Computing and Ambient Intelligence

References

1. http://www.softwaretestingclass.com/introduction-to-mobile-application-testing /
2. Tillmann, N., Schulte, W.: Stub-object generation with behavior. In Proceedings of the 21st IEEE/ACM international.
   Conference on Automated Software Engineering, 2006 September 18–22.
   Automated Software Engineering. IEEE Computer Society, Washington, DC, 2006:365-368.
3. http://www.microsoft.com/windows/NetMeeting/Corp/reskit/Chapter11/default.asp