Automating State to State testing of mobile applications to mobile platforms

Jonathan Sanders
Dr. Justice

ABSTRACT

This paper outlines a new way to test a devices states' affect on an application.

KEYWORDS

automata for mobile device testing,automated state testing

ACM Reference Format:
Jonathan Sanders and Dr. Justice. 2017. Automating State to State testing of mobile applications to mobile platforms. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A common problem with automated mobile software testing is testing mobile applications when the state of the device changes. Testing the different states of an application has been enhanced with tools such as Espresso and Robotium [1]. The problem with this issue is that often, the state of a device will cause a catastrophic failure of a feature within an application when the state of a mobile device changes; therefore, it is more expensive to not test device states against application states at all. Since the state of an application can be easily saved and the state of a device can be easily changed programmatically, a solution can be made that automatically tests application states with different device states. For applications being used in different device states, such as a mapping application used outside of network range or medical applications using sensors for monitoring functions, application to device state testing this becomes an extremely important issue. This author was once almost lost in the mountains because of one such issue.

"The field of mobile specific, black-box testing, however, remains thin." [7] That is the reasoning behind this endeavor to make easier testing tools that easily test the interaction of device states with applications states.

We have developed a Mobile application called "TADS" (Test Application to Device State) that uses a Mobile development UI tester to test the Mobile application against multiple states and sub-states of the device [9]. TADS is primarily concerned with identifying errors caused by changing device states and not as concerned with identifying why the state change caused a failure.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

As of the writing of this paper, there is not any research available on building out an automatic device state testing utility. In order to understand this project a few items must be known ahead of time.

Espresso, an android testing application, has an excellent tool for testing Mobile application UI's [6]. The tool records what is happening at a code level while a user performs different actions using the UI. This enables a working application to have a test automatically generated that can then be run at a later time which can be used to create a regression test suite. This actually enables states of the application to be tested without having to use any form of state machines or modeling. This tool called "Test Recorder" will be foundational to TADS.

3 RELATED WORK

There are several tools developed that can capture various information of an application and store said information for later tests. These tools can be leveraged to create interesting app states instead of simple object states. For instance the authors of [8] built a tool that automatically captures UI information as a user utilizes an application. The researchers with[4] made a tool that can store UI information and then auto generate oracles that then get scripted into an Espresso test script for later use as a test case. This tool could be used to run test cases as the DVCs execute as the state instead of static properties being evaluated.

Many others, such as [4] have made tools for recording tests that can be later executed in ways that are plaform independent which is a nice utility that we will not be concerned with in this work. Others have made unit level state testing models such as [5] in which the state of the application is tested by automatically building a model using a dynamic and static crawler and then running that created model against a verification algorithm. These researchers developed a way to use model testing to find security vulnerabilities [2]. Some have used machine learning to find the states of an application that are probably of value to test that have not been tested and make a model of those states for testing [3].

4 IMPLEMENTATION

The main purpose of this Application is to simplify testing of different device states with the application. There are preloaded "device state changes" (DSCs) that can be used in a test suite to test the app states that are chosen for testing. A DSC is a test case that starts with a device state then samples the chosen beginning app state, then changes the Devices state then sampling the app state again and evaluating if there were any exceptions thrown or errors detected. A few examples of DSCs are: airplane mode off, sample the app state, switch airplane mode to on, re-sample the app state to evaluate.

REFERENCES

[1] 2016. Top 10 Mobile Testing Tools. (Nov 2016). http://www.optimusinfo.com/top-10-mobile-testing-tools/

[2] G. Bai, Q. Ye, Y. Wu, H. Merwe, J. Sun, Y. Liu, J. S. Dong, and W. Visser. 2017. Towards Model Checking Android Applications. *IEEE Transactions on Software Engineering* PP, 99 (2017), 1–1. https://doi.org/10.1109/TSE.2017.2697848

[3] Wontae Choi, George Necula, and Koushik Sen. 2013. Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning. *SIGPLAN Not.* 48, 10 (Oct. 2013), 623–640. https://doi.org/10.1145/2544173.2509552

[4] M. Fazzini, E. N. D. A. Freitas, S. R. Choudhary, and A. Orso. 2017. Barista: A Technique for Recording, Encoding, and Running Platform Independent Android Tests. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. 149–160. https://doi.org/10.1109/ICST.2017.21

[5] Amin Milani Fard, Mehdi Mirzaaghaei, and Ali Mesbah. 2014. Leveraging Existing Tests in Automated Test Generation for Web Applications. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE '14)*. ACM, New York, NY, USA, 67–78. https://doi.org/10.1145/2642937.2642991

[6] Godfrey Nolan. 2015. *Agile Android.* Apress Disctributed to the Book trade worldwide by Springer Science+Business Media, Berkeley, CA New York.

[7] Fernando Paulovsky, Esteban Pavese, and Diego Garbervetsky. 2017. High-coverage testing of navigation models in Android applications. In *Proceedings of the 12th International Workshop on Automation of Software Testing*. IEEE Press, 52–58.

[8] F. Paulovsky, E. Pavese, and D. Garbervetsky. 2017. High-Coverage Testing of Navigation Models in Android Applications. In *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*. 52–58. https://doi.org/10.1109/AST.2017.6

[9] C. D. Turner and D. J. Robson. 1993. The state-based testing of object-oriented programs. In *1993 Conference on Software Maintenance*. 302–310. https://doi.org/10.1109/ICSM.1993.366932