Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 284 – 291

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Impact of execution modes on finding Android failures

Inês Coimbra Morgado[a,b,*], Ana C. R. Paiva[a,b]

[a]*Department of Informatics Engineering, Faculty of Engineering of University of Porto, Porto, Portugal*
[b]*INESC TEC Porto, Porto, Portugal*

Abstract

The iMPAcT tool combines the benefits of existing user recurring behaviour (User Interface Patterns) on mobile applications to facilitate the test automation of Android mobile applications. It uses an automatic exploration process combined with reverse engineering to identify the existing user interface patterns on a mobile application and then tests those patterns with generic test strategies (designated Test Patterns). The Test Patterns are defined in a catalogue that can be reused for testing other applications. However the results obtained by the iMPAcT tool depend on the exploration mode and on the order in which the test strategies are applied. This paper describes an experiment conducted to evaluate the impact of using different exploration modes and of changing the order by which UI patterns are searched and subsequently tested on the failures found and on the number of events fired.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Mobile Testing; UI Patterns; Reverse Engineering; Android; Test Automation; Case Study

1. Introduction

Smartphones have been gaining a strong participation in our daily lives. Furthermore, the number of mobile applications available has exceeded one million and the number of downloads has exceeded fifty billion[1].

This huge number of applications increases the rivalry among suppliers that need to ensure that their applications work correctly as thoroughly as possible if they want them to make a stand. Even though there are several approaches focused on test automation[2–6], the peculiarities of the mobile world, such as new development concepts, like activities, new interaction gestures and limited memory, make mobile testing a challenging activity[7,8]. The World Quality Report 2014-15[9] mentions that the greatest challenge for mobile testing is the lack of the right testing processes and methods, followed by insufficient time to test and the absence of in-house mobile test environments. Thus, it is extremely important to automate mobile testing.

Reverse engineering[10] is a technique used to aid in test automation as it provides information about the AUT. When reverse engineering a mobile application, dynamic techniques may be more appropriate than static ones due to the event-based nature of mobile applications: in this kind of applications most of the content is produced dynamically,

* Corresponding author. Tel.:+351225081400, fax:+351225081440
E-mail address: pro11016@fe.up.pt

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.127

e.g., the information presented on the screen depends on the order of the events previously fired, which is extremely difficult to analyse statically.

There are also some studies showing the usefulness of using patterns for mobile testing. In 2009, Erik Nilsson[11] identified some recurring problems when developing an Android application and the User Interface (UI) patterns that could help solve them. In 2013, Sahami Shirazi *et al.*[12] studied the layout of Android applications trying, among other goals, to verify if these layouts presented any UI Patterns. They concluded that 75.8% of unique combinations of elements appeared only once in the application. This study was conducted taking into consideration a static analysis of the layout and its elements. There is also some literature on the presence of UI Patterns on mobile applications, such as[13].

One of the main problems of dynamic reverse engineering is the dependence between the order in which the application is explored and the results obtained. The exploration may be random or guided, *i.e.*, follow an exploration algorithm, such as depth-first.

The iMPAcT tool[14] combines the benefits of existing UI Patterns and reverse engineering to ease mobile test automation. The tool uses dynamic reverse engineering to detect the presence of UI patterns and then tests them using generic test strategies, called Test Patterns (more details in Section 3). The results obtained by the tool depend both on the order in which the different events are executed and on the order the different patterns are identified and tested. This paper reports the results obtained by the iMPAcT tool when using different exploration modes and different Test Patterns testing orders.

The remaining of the paper is structured as follows. Section 2 presents some related work. Section 3 describes the approach implemented in the iMPAcT Tool. Section 4 presents the case study and corresponding results. Section 5 presents the drawn conclusions.

2. Related Work

Model based testing (MBT) approaches generate test cases from models according to coverage criteria. Even though MBT generates test cases automatically, the effort invested in building the model should not be neglected. To reduce this effort it is possible to use reverse engineering approaches to extract part of an existing application model and work from there.

Software reverse engineering has long been a field of research. There are several approaches that extract models from desktop[15–17] and web[18–20] applications. Regarding the mobile world, to the best of our knowledge, there is no approach focusing Windows Phone applications and only a handful of approaches dealing with iOS application, in the last years.

Even though reverse engineering approaches can be static (when they analyse the source code), dynamic (when they analyse the application at run time) or a combination of both (hybrid), the event-based nature of mobile applications makes the dynamic and hybrid approaches more common. Nevertheless, there are some, like Batyuk *et al.*[21] who identify possible security vulnerabilities like unwanted user access, by applying static approaches.

Regardless of the techniques used, the purpose of reverse engineering in the context of mobile applications is to obtain a model of the application and/or to test it. The work of Yang *et al.*[22] is an example of the first as it presents a hybrid approach: an initial static phase identifies the possible events to be fired and a second dynamic phase explores the application by firing those events and analysing their effects on the application. An example of the latter is the work of Amalfitano *et al.* in 2012[7] which is similar to the exploration phase of the approach presented in this paper and in 2013[23], in which they generate test cases but follow a dynamic approach with reflection and code replacement techniques.

Mobile application testing (or mobile testing) has been gaining interest by researchers because it presents additional challenges when compared to the testing of other types of applications, such as, web or desktop[8].

Mobile testing can be performed automatically and the market already offers several automatic options for test case execution, including the two official ones for Android: Espresso[1] and UI Automator[2]. The main difference between

[1] https://developer.android.com/training/testing/ui-testing/espresso-testing.html
[2] https://developer.android.com/training/testing/ui-testing/uiautomator-testing.html