1J-05

Mobile Application Testing Focusing on External Events

Siena Yu† Shingo Takada‡
Keio University

1. INTRODUCTION

Mobile application testing is not as simple compared to other applications such as web applications. User inputs in web applications are limited to keyboard and mouse only. However, the presence of hardware components in mobile devices, such as phone capability, GPS, and hardware sensors, affects the testing of mobile applications and makes it complicated.

In this paper, we implemented our approach on test case generation for Android applications, which was presented in our previous paper [7]. Our approach focuses on external events and considers both explicit and implicit types of external events, which are events that are handled and not handled by the mobile application code, respectively.

An implicit event is an event that may occur which is not explicitly handled within an Android application code. Such event should also be tested. For example, an application may have code to handle the case where GPS becomes available, but if it does not have the code to handle the case where GPS becomes unavailable, what happens? Our work focuses on such cases.

The remainder of this paper is structured as follows. Section 2 discusses the related works. Section 3 then describes our approach. Lastly, section 4 presents the conclusion.

2. RELATED WORKS

There are several works on testing mobile applications, but most of them focus on GUI inputs. Amalfitano, et al. [3] created a tool called AndroidRipper for automatically testing Android applications based on GUI Ripping. Anand, et al. [4] and Jensen, et al. [5] used concolic testing to generate event sequences in their tools ACTEve and Collider. These works are successful in automatically generating test cases for mobile applications. However, they did not consider the external events that may affect the mobile application.

There are few works on testing mobile applications focusing on external events. Amalfitano, et al. [2] and Morgado, et al. [6] used dynamic analysis of the mobile application in their proposed approaches. Both of them manually defined patterns of events to be used in generating test cases. Adamsen, et al. [1] proposed an approach by using existing test suite and executing them in adverse conditions. They used event sequences that do not affect the outcome of the tests and injected these into test cases.

However, these works only considered explicit events, which are events that are handled by the mobile application. There are some events that a mobile application should handle and ensure that it will not crash even if these events are not handled by the code. We refer to these events as implicit events.

3. OUR APPROACH

In this section, we will describe the architecture of our proposed approach as shown in Figure 1. The system is composed of six major parts: (1) Event Repository, (2) Event-Pattern Repository, (3) Event-Pattern Generator, (4) Static Code Analyzer, (5) Basic Test Case Generator, and (6) Augmented Test Case Generator.

The Event Repository stores the different external events and their corresponding categories. These are manually defined. Some examples of external events are shown in Table 1.

These events can be combined to form different event-patterns that can be used to create new test cases. Some example of event-patterns are shown in Table 2[1]. The Event-Pattern Generator is responsible for generating these event-patterns and storing it in the Event-Pattern Repository. Events from the same category are combined to form event-patterns. A condition can be added to each event to remove unnecessary or impossible combinations. Example conditions are (1) “Idle cannot be an initial event and only comes after Ringing or Off-hook”, (2) “Off-hook cannot be an initial event and only comes after Ringing”, and (3) “Ringing cannot be a last event”. Given these conditions, we can generate event-patterns P1 and P2 for category “Phone Call” as shown in Table 2.

The Static Code Analyzer uses static analysis of the Android code to detect the external events that the mobile application can sense and react to. Then, the

“Mobile Application Testing Focusing on External Events”
† Siena Yu・Keio University
‡ Shingo Takada・Keio University

[1]Note that we have elided “Becomes” from Table 2, e.g., “Ringing” denotes “Becomes Ringing”. “Becomes” will be elided in the rest of this paper.

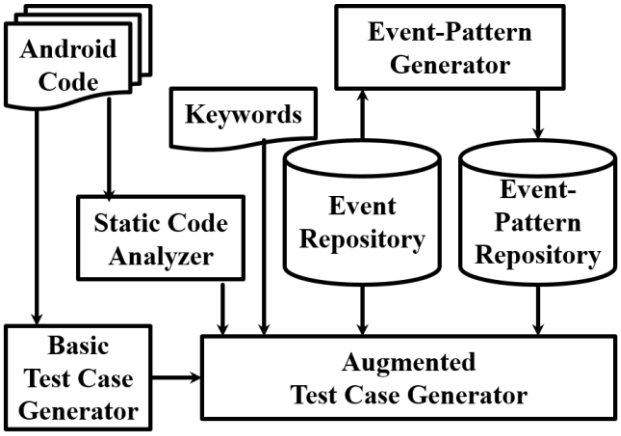

Figure 1: Architecture

events detected are grouped into categories and these categories will be used to retrieve event-patterns from the repository. This will ensure that all events from the same category will be considered in creating test cases, which will cover both the explicit and implicit events. For example, the event "GPS Available" may be handled by the code (explicit event), but the event "GPS Unavailable" may not (implicit event).

Users can supply events or categories of events in the Keywords to ensure these events are considered in generating test cases. This enables the generation of test cases for implicit events. For example, even though an app may not explicitly handle phone call events, the user may want to generate test cases for phone call events. Our approach also recommends categories of events to be tested. Users can choose from these recommendations. This will also generate test cases for implicit events.

There are two Test Case Generators. First, the Basic Test Case Generator generates test cases in conventional manner, which focuses on GUI or internal events, based on Android Code information. Then, the second, the Augmented Test Case Generator, incorporates the event-patterns to the test cases generated by the Basic Test Case Generator using information from Keywords and Static Code Analyzer. Each pattern can be inserted at the start, middle, or end of a Basic Test Case. For example, we have the basic test case: (1) "Click Settings" and (2) "Check Display Location"; and the event-pattern: (A) "Ringing" and (B) "Idle". The augmented test cases would be (TC1) "A, B, 1, 2", (TC2) "1, A, B, 2", and (TC3) "1, 2, A, B".

We show an example of generating test cases. The user specifies "Phone Call" in Keywords and the Static Code Analyzer detects the event "GPS Available" (E4 in Table 1). Based on the information from the Keywords, the implicit events are E1, E2, and E3. Then, based on the information from the Static Code Analyzer, the explicit event is E4 and the implicit event is E5. The Augmented Test Case Generator will then retrieve

Table 1: External Events

Category	Event	ID#
Phone Call	Becomes Idle	E1
	Becomes Off-hook	E2
	Becomes Ringing	E3
GPS	Becomes Available	E4
	Becomes Unavailable	E5

Table 2: Event-Patterns

Category	Event-Pattern	ID#
Phone Call	Ringing, Off-hook, Idle	P1
	Ringing, Idle	P2
GPS	Available, Unavailable	P3
	Unavailable, Available	P4

event-patterns from the detected categories of events, which are "Phone Call" and "GPS". The event-patterns from these categories will be incorporated to the test cases generated by the Basic Test Case Generator to form new test cases that consider external events.

4. CONCLUSION

In this paper, we proposed an approach for test case generation for Android applications, which focuses on external events. Our approach considers both the explicit and implicit types of external events.

5. REFERENCES

[1] C. Q. Adamsen, G. Mezzetti, and A. Møller. Systematic execution of Android test suites in adverse conditions. In Proc. of ISSTA'15, pages 83–93, 2015.

[2] D. Amalfitano, A. R. Fasolino, P. Tramontana, and N. Amatucci. Considering context events in event-based testing of mobile applications. In Proc. Of ICST'13, pages 126–133, 2013.

[3] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, and A. M. Memon. Using GUI ripping for automated testing of Android applications. In Proc. of ASE'12, pages 258–261, 2012.

[4] S. Anand, M. Naik, M. J. Harrold, and H. Yang. Automated concolic testing of smartphone apps. In Proc. of FSE'12, 2012. 11 pages.

[5] C. S. Jensen, M. R. Prasad, and A. Møller. Automated testing with targeted event sequence generation. In Proc. of ISSTA'13, pages 67–77, 2013.

[6] I. C. Morgado, A. C. R. Paiva, and J. P. Faria. Automated pattern-based testing of mobile applications. In Proc. of QUATIC'14, pages 294–299, 2014.

[7] S. Yu, and S. Takada. External Event-Based Test Cases for Mobile Application. In Proc. of C3S2E'15, pages 148-149, 2015.