ACAT: A Novel Machine-Learning-Based Tool For Automating Android Application Testing

Ariel Rosenfeld[1,†], Odaya Kardashov[2], and Orel Zang[2]

[1] Dept. of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot, Israel
[2] Dept. of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
[†] Corresponding author, arielros1@gmail.com

Abstract. Mobile applications are being used every day by more than half of the world's population to perform a great variety of tasks. With the increasingly widespread usage of these applications, the need arises for efficient techniques to test them. Many frameworks allow automating the process of application testing, however existing frameworks mainly rely on the application developer for providing testing scripts for each developed application, thus preventing reuse of these tests for similar applications. In this demonstration, we present a novel tool for the automation of testing Android applications by leveraging machine learning techniques and reusing popular test scenarios. We discuss and demonstrate the potential benefits of our tool in an empirical study where we show it outperforms standard methods in realistic settings.

Keywords: Android Application Testing, Mobile Testing Automation, Activities Classification, Demonstration

1 Introduction

Mobile devices become a key component in our lives, with more than five million applications developed so far [1], making them the main productivity feature of these devices. As mobile devices become more popular, arise the need for efficient techniques for testing their applications. The large fragmentation of the Android market, as well as the diverse set of scenarios in which a mobile application can be used, make testing new applications an expensive, time-consuming and a complex process. Recently, test automation became the standard, with many solutions and frameworks that allow automating the process of application testing [2]. The main limitation of these frameworks is that tests are hand-coded for *each application and scenarios*, and each new application or new functionality requires spending many resources to reuse these tests.

In this demonstration, we present a novel tool for automatic testing of Android applications in order to find as many functional bugs as possible. Our tool is based on the premise that different activities in an Android application share a similar structure. In order to use this similarity to our benefit, we use machine learning techniques to classify each activity in the application into one of

seven pre-defined activity types. For each classified activity, we can run specific tests, at user interface level, that were coded to utilize the fact that we know its general structure and desired behavior. We have implemented this approach and developed an add-on in the Java programming language for the TestProject[1] test automation framework. The platform includes hundreds of add-ons for Web, Mobile and API testing which are freely available to anyone wishing to accelerate the automation development project. Our developed add-on is named ACAT, standing for "Activities Classification for Application Testing". The add-on will be available to install via TestProject Add-ons store.

To evaluate our add-on, we conducted an experiment in which we ran it on different applications while evaluating their performance. We found that the ACAT add-on shows great ability, compared to the standard random-testing method, in exploring the application's state space and testing its key components without prior knowledge about the application. This lets the developer focus on the development of the application and not on writing tests. The use of machine learning in application testing tools is, to the best of our knowledge, a novel method which has yet to be fully explored.

2 Demonstration

In this demonstration we are going to illustrate a common scenario in testing an Android application with the ACAT add-on. Before executing the ACAT, the developer provides basic information using a textual input file which includes technical parameters which are required for the TestProject framework to communicate with the mobile device. Then, the developer can provide specific information in the input file, such as a username and a password, which the add-on will later utilize during the test in the correct context.

During the test, for each new reached activity in the application, the ACAT classify it using our machine learning model into one of seven pre-defined activities types. Then, it executes a series of test cases designed for the predicted activity type. The ACAT keeps classifying activities and deploying tests until the main activity of the application has been classified and tested (e.g., mail activity, browser activity, etc) or when its configured time is up.

Finally, after executing the test, the ACAT creates a textual report file which includes comprehensive information about the test. This includes some various statistics about the test, such as the number Of discovered activities and a list of all the actions executed by the add-on which allows the developer to track the steps of the test. The most important part of the report is the description of all the bugs that were discovered during the test, either technical real-time crashes or logical defects which reflect in the application's user interface. An example for the bugs section of this report can be found in Figure 3. Figures 1 and 2 depict the results of running the ACAT on Android's default mail application while explaining about the underlying process.

[1] `http://testproject.io`

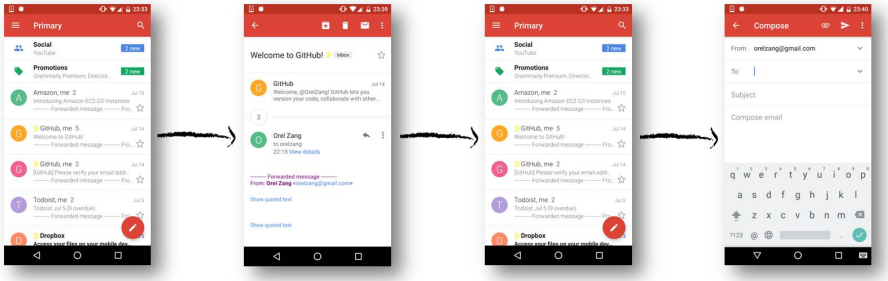

Fig. 1: The activity was classified as a mail activity. Thus, the ACAT executes the mail activity test cases. It scrolls through the inbox mails and opens a random one from the list. Then, it returns to the inbox screen and opens the compose mail activity.

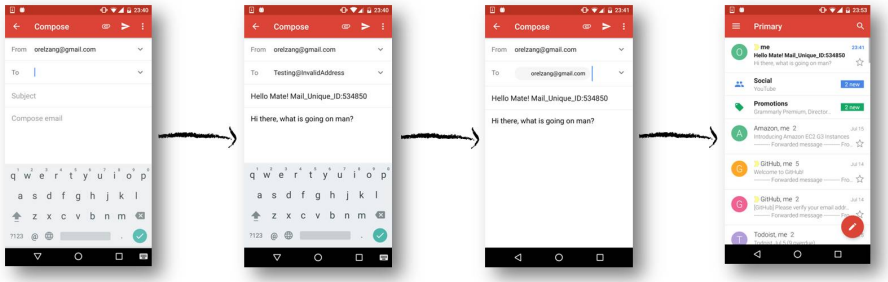

Fig. 2: The ACAT verifies that a user cannot send a mail without a recipient address and with an invalid address. Finally, the ACAT verifies that a user can send a valid mail through this activity by sending a mail to the same user while filling in a randomly generated ID in the mail's subject. Then, the ACAT refreshes the inbox list and searches for a mail with the ID that was sent before.

```
Application Crashes:

    .ui.MailActivityGmail - Passed - No Crashes Have been found.

    .ComposeActivityGmail - Passed - No Crashes Have been found.


Activities Sub Tests:

    .ui.MailActivityGmail - Classified As Email Activity. Activity Test Passed -
    All Sub Tests Have Been Passed:
        Browsing through the inbox mails - Passed
        Opening a mail from the inbox mails list - Passed
        Scrolling an email content from inbox up and down and return to inbox list - Passed
        Opening compose new mail form - Passed

    .ComposeActivityGmail - Classified As Compose Email Form Activity. Activity Test Passed -
    All Sub Tests Have Been Passed:
        Typing in the email subject and body - Passed
        Sending an email without recipient address - Passed
        Sending an email with invalid recipient address - Passed
        Sending and receiving a valid email - Passed
```

Fig. 3: An excerpt from the report which was generated after this test.

3 Evaluation & Discussion

We evaluate our add-on against the Android Application Monkey tool [3]. With the purpose of demonstrating the power of the ACAT, we designed a novel ex-

periment which focuses on applications logical bugs. These bugs are related to the application's logic, meaning unwanted behavior in the application's functionality, as opposed to real-time application crashes which are caused by uncaught exceptions thrown in the code.

We use 2 open source Android applications in which we artificially "plant" various logical bugs. While examining the experiment's results, which we will present at the demonstration as well, we can identify 2 major trends: 1) The ACAT was able to classify correctly the 3 unseen activities; 2) The ACAT managed to discover all of the "planted" bugs while the Android Monkey discovered none.

These trends show an interesting phenomenon. While the Android monkey was not able to detect a single logical bug, the ACAT discovered all of the various bugs implemented in the source code of the applications, *as well as a bug that already existed in the original code*. This is contributed to our activities classification method, which enables this add-on the power to tests an activity against its expected behavior. In addition, our experiment demonstrates that testing an application by a series of single case tests, designed for each activity at its own, may provide an advantage compared to the standard testing of applications which considers the application as a whole unit.

4 Conclusions

This paper introduces a novel add-on for testing Android applications using machine learning techniques. We have tested our add-on on different applications, demonstrating its advantage against other popular Android applications testing tools such as the Android Monkey. The ACAT add-on is shown to find more logical bugs in an application, which opens the possibility for developing more sophisticated testing tools. We are currently working with TestProject in order to integrate the ACAT add-on for the TestProject framework, utilizing their database of thousands of mobile applications patterns. The ACAT add-on will be available to install via TestProject Add-ons store. For more technical details about the approach of the tool see the full publication [4].

References

1. "Number of apps available in leading app stores as of march 2017." `https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/`.
2. J. Gao, X. Bai, W.-T. Tsai, and T. Uehara, "Mobile application testing: a tutorial," *Computer*, vol. 47, no. 2, pp. 46–55, 2014.
3. S. R. Choudhary, A. Gorla, and A. Orso, "Automated test input generation for Android: Are we there yet?," in *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pp. 429–440, IEEE, 2015.
4. A. Rosenfeld, O. Kardashov, and O. Zang, "Automation of Android Applications Testing Using Machine Learning Activities Classification," *ArXiv e-prints*, Sept. 2017.