International Journal of Open Source Software and Processes
Volume 11 • Issue 1 • January-March 2020

Mutation Testing to Evaluate Android Applications

Ahmad A. Saifan, Information Systems Department, Faculty of IT, Yarmouk University, Irbid, Jordan

Ahmad Adnan Alzyoud, Information Systems Department, Faculty of IT, Yarmouk University, Irbid, Jordan

ABSTRACT

Android is an operating system source which offers flexibility and support for most mobile applications, and easy access to social networks. It is important to understand the complexity of design, development, implementation, and testing of Android apps. A number of challenges may be faced in testing android applications, including the lack of testing processes and methods, testing experts being unavailable, poor in-house testing environment, and time restrictions. Mutation testing is a fault-based testing technique, applied by generating mutants and running the application with these mutants to analyze the killed and equivalent mutants. We defined a set of mutation operators according to the features of android applications: apps with content sharing, apps with multimedia, apps with graphics, and apps with user location and maps. We identified 42 mutation operators. In addition, we implemented a new tool, "µ-Android," which automatically generates mutants and retrieves results to prove the efficiency of the test cases and enable the new operators.

KEYWORDS

Android, Mdroid+, Mutation Testing, µ-Android

1. INTRODUCTION

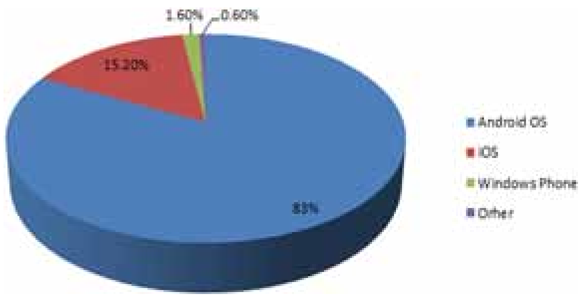
Android applications have been adopted significantly by many manufactures. In the third quarter of 2016, Android operating systems dominated the smart phone market with 83% of sales, and over 341.5 million units were shipped. Moreover, based in a study conducted by International Data Corporation (IDC, 2016), there are around 1.7 million android applications available on the Google Play store. Figure 1 represents the percentage of smart phone operating systems. The proliferation of Android applications, and continued reliance on Android devices in everyday life, raises the challenge of ensuring the quality of these applications.

The testing of Android applications presents certain challenges. Developing new android applications is expensive due to the variety of device types, and the applications developed must adapt to a variety of manufacturer specifications. Testing these applications on each device and ensuring compatibility and functionality is also time-consuming and involves considerable effort. In addition, the same application may not respond in the same way on all android devices. Also, the range in variety of android operating system versions may raise the problem of incompatibility. Finally, Android applications may violate some privacy and security concerns where Google does not approve third party android applications before they go to market, and personal information, activities, and statuses collected from different android applications may be used in an inappropriate manner.

Several approaches have been proposed in the literature to test android apps but they do not reach actual practice such as (Amalfitano et al., 2012; Amalfitano et al., 2011; Azim & Neamtiu, 2013;

DOI: 10.4018/IJOSSP.2020010102

Figure 1. Apps percentage in markets



DeMillo, Lipton, & Sayward, 2013; Hu & Neamtiu, 2011; Mahmood, Mirzaei, & Malek, 2014). However, some other research in the literature proposed mutation testing tool such as (Deng, Offutt, Ammann, & Mirzaei, 2017; Linares-Vásquez et al., 2017). Despite the availability of these approaches, the field of android apps testing is still very much under development; as evidenced by limitations related to the mutation operators used to cover the extensive feature of android apps such apps with content sharing; apps with multimedia; apps with graphics; and apps with location and maps. In this paper, we focus on ensuring that the Android applications are reliable before they are released. The aim of this paper, which we share with previous work, is to define new mutation operators suitable to detect hidden faults in android applications, by relying on an empirical foundation. So, our mutation operates are complement with other mutation operators used in the literature. They are used to cover some features that are not covered by previous work, i.e. they are used to detect hidden faults that cannot be predicted by the other mutation operators. Specifically, we use mutation testing as a technique to test android apps. Several approaches in the literature have been used to generate test cases from given android apps (Choudhary, Gorl, & Orso, 2015; Machiry, Tahiliani, & Naik, 2013), such as branch coverage, statement coverage, method coverage, etc. Mutation testing is a good technique to compare with because of its effectiveness and efficiency as has been proven before (Kintis et al., 2016; Praphamontripong, & Offutt, 2010). Rene Just et. al. (2014) showed that mutation testing is better in prediction the rate of real faults than code coverage. Moreover; mutation-based approach to automatically generating test suites is promising. Mutation testing can helps in finding the hidden faults with maximum code coverage. It also, can be used to check the quality of test cases that are used to test a system under test.

The main goal of this research is to answer the following questions: Can we suggest a new set of mutation operators for Android applications? What are these operators? How can we measure these mutation operators' effectiveness in terms of their ability to inject faults?

In this paper, we introduce a set of effective mutation operators that can be used to detect the faults in android applications. These mutation operators are categorized into four types based on a specific fault feature of android application: apps with content sharing; apps with multimedia; apps with graphics; and apps with location and maps.

This paper is structured as follows: Section Two provides the necessary background, briefly outlining the mutation analysis process, and describing an Android operating system and its application. Related work is reviewed in Section Three. Section Four defines a set of novel mutation operators that are organized into four levels based on the specific fault features of Android apps. Section Five describes the implemented mutation tool which is the μ-Android. The evaluation process and applications used are discussed in Section Six. Finally, the conclusion of the research and recommendations for future work are presented in Section Seven.