УДК 004.75

Ю. М. Виноградов,
М. О. Байдиков

ПИТАННЯ ЯКОСТІ МОБІЛЬНИХ ДОДАТКІВ НА ОСНОВІ ЇХ ТЕСТУВАННЯ

MOBILE APPLICATIONS QUALITY ISSUES BASED ON TESTING

У статті розглядається питання підвищення якості мобільних додатків. Для цього використовується повноцінне тестування, яке включає в себе тестування окремих модулів додатків, тестування поведінки всього додатку та тестування поведінки додатку під великим навантаженням. Для тестування використовуються підходи Given-When-Then, Behavior Testing, Fuzz testing та Performance Testing.

Ключовіслова: тестування, Given-When-Then, Behavior Tests, Fuzz tests, Performance Tests.

The paper deals with the issues of increasing quality of mobile applications using testing. As main testing types used Given-When-Then, Behavior Testing, Fuzz testing and Performance Testing. All these testing types cover testing separate modules of application, testing the behavior of whole application and testing application under unexpected cases of usage.

Key words: testing, Given-When-Then, Behavior Tests, Fuzz tests, Performance Tests.

Target setting. Due to growing number of mobile applications appeared last time, the question of their quality has become the main thing to think about.

Actual researches and issues analysis. During last time some big companies made this topic one of the most discussed at their conferences, but it’s still isn’t covered enough.

Uninvestigated parts of general matters defining. There is a huge lack of works, describing Fuzz and Performance testing of mobile applications. Moreover, even usual Given-When-Then and Behavior tests remain investigations. And the biggest gap in all these investigations in connections all this different testing types into one stable ecosystem.

The research objective. The purpose of this paper is to investigate mobile applications testing types and their combination. As a solution, the article will focus on creating one single ecosystem of different testing techniques which can show as much as possible bugs of mobile applications. This approach will help all the developers to increase the quality of their applications and reduce the number of application crashes.

The statement of basic materials. In general, testing is a process of writing some code, which runs app modules or the whole app, performs some actions, collects results and checks their equality to the expected result, provided by developers [1]. Testing can be divided into several types. The main testing types, applicable to the most popular mobile operating systems, iOS and Android, are:

1. Unit testing - tests individual parts of code for correctness, using an automated test suite. A good unit test assures the functionality expecting out of the unit is correct. A good unit test is repeatable, fast, readable, independent and comprehensive.

2. Given-When-Then testing - similar to unit testing, but declares strong format of test, which must be inherited to achieve the proper result.

3. UI Testing - takes a flow that the user might follow and ensures it produces the right output. Can be performed both on real devices or emulators and simulators. Given the same state, the UI test always must produce the same output. The input to the test in the user clicks and the output is the screen.

4. Behavior testing - very similar to UI testing, but instead of testing the whole user flow, test small user actions.

5. Fuzz testing - unit tests and UI tests are user to ensure that the expected output happens when the expected input is used. However, users can be unpredictable and can do unexpected things. For such cases comes the fuzz testing. It introduces a random stream of events into app and records result. It's a stress test which simulates random presses on the screen. This is very similar to UI testing, but it has no input and it doesn't check some defined user flow.

6. Performance testing- gathers performance metrics of the app to ensure that application doesn't take up device resources. This metrics include network usage, memory usage, battery drain.

7. Continuous Integration is a tool to collect all the above techniques into one flow. It is a command line tool, which is hosted on external server and observes application repository. When a developer performs commit, this tool takes the application from this commit and performs all the tests. If all of them are success, then the tool pushes all the changes to remote. In other case, it notifies the developer, that something is wrong with the app and must be fixed before pushing [2].

Table 1 illustrates all the testing types and native frameworks, used on iOS and Android platforms for these types of testing [3].

Table 1

Testing types and native frameworks

Testing type	iOS native framework	Android native framework
Unit testing	XCTest	JUnit
Given-When-Then testing	-	-
UI testing	XCUITest, Xcode UI Test Recorder	Espresso, Espresso Test Recorder
Behavior testing	-	-
Fuzz testing	-	-
Performance testing	Instruments (only for collecting data)	Android Monitor (only for collecting data)
Continuous Integration	-	-

As the table shows, native testing frameworks on each platform don't provide developers with all the required tools to performs full mobile application testing. Moreover, the don't provide anything to perform Given-When-Then testing and are not suitable for Behavior Testing [4].

The correct testing frameworks. The correct testing frameworks must include all the required tools to perform all the given testing types. To achieve these requirements, were developed next frameworks.

Table 2

Developed frameworks for testing

Testing type	iOS framework	Android framework
Unit testing	Quick+Nimble	JUnit
Given-When-Then testing	SwityGWT	KotlinGWT
UI testing	XCUITest, Xcode UI Test Recorder	Espresso, Espresso Test Recorder
Behavior testing	Quick+Nimble	KotlinBDD
Fuzz testing	UI AutoMonkey	monkeyrunner
Performance testing	New Relic Mobile	New Relic Mobile
Continuous integration	Jenkins	Jenkins

Experiments. To check developed frameworks, was developed a small notes application. After that, one it's copy was covered by 30% using native frameworks of each platform. The real usage of this app showed ten crashes per week for one thousand users. The result seems to be not bad, but testing app with developed frameworks leads to 50% code coverage and only one crash per week for one thousand users. The difference is big even at this point, but scaling app to ten thousands users leads to one hundred crashes per week for the version, tested by native frameworks, while the version, tested with developed frameworks achieved the result of ten crashes per week.

Conclusion. The paper has demonstrated the ability of proper testing to decrease number of crashes of mobile application up to ten times even on the small audience. The developed frameworks can be used with any application and don't required lots of time to develop all the test types. Using such combination of frameworks together with continuous integration produces qualitative results.

There are still several directions for future work. One is to develop frameworks for UI Testing for both platforms to detect different UI bugs of applications. Another is to increase simplify setup of all these frameworks and continuous integration. These changes will definitely improve the result.

References

1. Daniel Knott (2015). Hands-On Mobile App Testing: A Guide for Mobile Testers and Anyone Involved in the Mobile App Business. Addison-Wesley Professional; 1 edition, May 28, 2015.

2. Paul Blundell, Diego Torres Milano (2015). Android Application Testing. Packt Publishing - ebooks Account, April 30, 2015.

3. Vijay Velu (2016). Mobile Application Penetration Testing. Packt Publishing - ebooks Account, March 18, 2016.

4. Mobile Testing: Complete Guide to Test your Mobile Apps. Update date: February 15, 2018. URL: https://www.guru99.com/testing-mobile-apps.html (application date: April 29, 2018).

ДОВІДКАПРОАВТОРІВ

Виноградов Юрій Миколайович – старший викладач, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Vinogradov Yurii – associate professor, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

БайдиковМиколаОлександрович - студент, кафедраобчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Baidykov Mykola - student, Department of Computer Engineering, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

Email: nbaidikoff@gmail.com

Ю. М. Виноградов, М. О. Байдиков

ПИТАННЯ ЯКОСТІ МОБІЛЬНИХ ДОДАТКІВ НА ОСНОВІ ЇХ ТЕСТУВАННЯ

Актуальність теми дослідження. Проблема якості мобільних додатків стала актуальною в останні дні у зв’язку з різким збільшення їх кількості та зростанням попиту. Це спричинило появу великої кількості додатків за короткий період часу без належного тестування. Крім того, самі розробники середовищ створення додатків не дають достатньо ресурсів для написання якісних тестів. Дана стаття присвячена створенню оптимального набору ресурсів, які дозволять проводити повноцінне тестування додатків.

Постановка проблеми. Відсутність якісних ресурсів для повноцінного тестування мобільних додатків та застосування автоматизованого тестування.

Аналіз останніх досліджень і публікацій. Протягом останніх років з’являється все більше праць присвячених тестуванню мобільних додатків. Проте всі вони зосереджуються на якомусь одному методі тестування (найчастіше на Юніт тестуванні), у той час як матеріалів про загальне тестування зі застосуванням декількох різних методів майже немає.

Виділення недосліджених частин загальної проблеми. Дана стаття присвячена аналізу існуючих рішень та створенню нових для повноцінного тестування мобільних додатків. Дослідження сфокусовано на вивченні осонвних методик тестування, їх переваг та недоліків.

Постановка завдання. Завдянням є створити оптимальний набір інструментів для всеохоплюючого тестування мобільних додатків на двох сучасних платформах, iOS та Android.

Викладення основного матеріалу. Проведено аналіз стандартних засобів тестування для iOS та Android, виділено їх найголовніші недоліки. Описано вимоги до засобів тестування, які повинні забезпечити можливість повноцінного покриття тестами будь-якого додатку. Створено необхідні засоби.

Висновки. Проаналізовано створені засоби тестування на невеликому додатку з різною кількістю користувачів. Наведені результати експериментів та аналіз вище вказаних кроків.

Ключові слова: тестування, Given-When-Then, BehaviorTests, Fuzz tests, PerformanceTests.