Designing an Android continuous delivery pipeline

Milena Zachow

Fachbereich 3: Information & Kommunikation
University of Applied Sciences Flensburg
Kanzleistraße 91-93
D-24943 Flensburg
milena.zachow@fh-flensburg.de

Abstract: Mobile applications (apps) are increasingly popular and run on a wide range of different operating systems and devices. Fragmentation is one of the differences between mobile apps and web- or desktop based applications and presents a challenge in delivering high quality apps. Automated testing can help to overcome it. This paper presents a case study on designing a continuous delivery pipeline for an Android app focused on simple setup.

1 Introduction

The use of mobile devices grows significantly. Mobile apps are different from traditional web- or desktop based applications in many respects. They run on highly fragmented devices and operating systems, use a variety of inputs from user and environment (e.g. sensor inputs, speech or gestures) and have limited resources (e.g. CPU, memory and battery power).

In web- and desktop continuous delivery is an important topic. Automation is necessary to ensure quality; the automated build and test steps are called *continuous delivery pipeline* [HF10]. In mobile app development continuous delivery and test automation culture is different [Ko15].

This paper describes the development of the mobile Android app *MedTabImager* that visualizes medical CT/MR slice images. *MedTabImager* is made of roughly 20k LOC. Rendering is based on OpenGL ES. The unit test code coverage varies between 39% in some packages (e.g. configuration or util) and single-digit numbers in in view-related packages (e.g. components). 2 developers implemented 30 user stories in a 9-month period (part time). Instrumentation tests exist for some of the user stories. Although in the medical domain software quality is important and tests exist, the developers did not set up a continuous delivery pipeline. That leads to the question:

RQ1: What are the challenges in setting up a continuous delivery pipeline for Android apps?

The *MedTabImager* developers set up a continuous integration (CI) server, but did not use it for automated testing. The reasons were technical challenges in making the CI work with the emulator or real devices. After spending a certain amount of effort (see Tab. 1) without satisfying results the developers gave up on the task.

Therefore one of the challenges in setting up a continuous delivery pipeline for apps is:

C1: Minimize the effort of the setup

There are other challenges as well. To name a few: the Android emulator is slow and unstable. UITests on the OpenGL level are hard to implement. Some sensor data is difficult to mock. This paper focuses on minimizing the pipeline's setup effort.

2 Related work

Continuous delivery is a topic in research [Fe13] but is often focused on building large web-based systems (e.g. at companies like Facebook or Amazon).

In the context of mobile apps, especially Android, automated testing is widely discussed in academic research and industry. Google just recently added full support of unit testing in Android studio (an experimental support exists since Version 1.1[1]). They also offer different solutions for testing Android lifecycle code and integration tests. For UI testing Google's current solution is a tool called UIAutomator[2]. Additionally, lots of commercial and free testing tools exist.

There have been many studies on testing techniques for mobile app development, especially Android. Tools for automated test case generation have been proposed as well as different approaches to automate UI testing with the capture and replay approach [Ch14]. Cloudbased solutions or *Testing as a Service* (TaaS) are widely discussed in both industry and research, judging by extensive research [Ga14]. Test automation is possibly the only way to deal with the continuing fragmentation [KK13]. However, in the mobile app development context the level of test automation is quite low. Studies on test coverage in open source Android apps suggest that a majority of the apps studies do not have any tests cases at all – nearly 86% [Ko15]. Automated testing does not seem to be widely accepted by Android developers.

The technical challenges to achieve continuous delivery on top of automated tests for mobile apps are seldom addressed in research. Continuous delivery of mobile applications is studied but with a focus on the process [Kl15].

[1] http://tools.android.com/tech-docs/unit-testing-support
[2] http://developer.android.com/tools/testing/index.html

3 Minimize setup effort with TaaS

The *MedTabImager* developers aimed at following steps for their automated build: compile and package the app (using the build management tool Gradle), run unit tests, run UI tests, send an apk file to beta users and deploy to a marketplace. They failed to manually set up UI tests on a continuous integration server (Jenkins) in a reasonable amount of time.

When introducing a TaaS approach to the project only the steps *build and run unit tests* were executed on the local CI server (see Figure 1). Since all unit tests run on the JVM no further setup time was required.

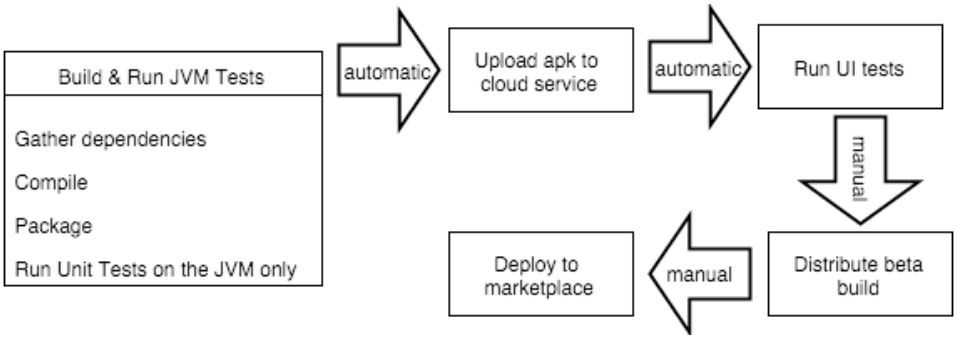

Figure 1: continuous delivery pipeline on a CI server based on TaaS

All UI Tests were implemented using the open source tool Appium[3]. Appium works on different mobile platforms and is widely supported by mobile test cloud services. The developers wrote tests and tested against their local emulator or device. Instead of setting up emulators on the local CI, the UI tests were run against a cloud service (Sauce Labs). Setting up Sauce Labs on the CI server with a specific plugin did not require much effort (see Tab. 1). A successful Gradle build (and unit test run) automatically triggered the UI tests on Sauce Labs.

Despite the automated tests, manual testing before a release was necessary. Not all user stories were covered by UI tests. To automate the app distribution a step was added to the pipeline: distribution of beta builds. Again, a cloud-based service was used to distribute the app (e.g. provide a download link and notify testers by email, once a new version is released). There are several services on the market. TestFairy[4] was used for the *MedTabImager*.. The build step was triggered manually, since beta testers were not required to test every development build.

For the *MedTabImager* the TaaS solution required less setup effort and worked fairly well. Tab. 1 compares setup times (roughly estimated by the developers) for the TaaS

[3] http://appium.io/
[4] https://testfairy.com/

and for local CI server solution (that did not work properly afterwards and was suspended).

Task	Hours manual setup (approximately)	Hours cloud service setup (approximately)
Basic setup	4	1
Headless emulator	8	-
Sensor input virtualization	4	-
Automated beta tests	4	1
Automation, bug fixing	4	1

Tab. 1: Comparison manual setup / cloud service setup

4 Summary & Conclusion

This paper presented a case study in which continuous delivery of an Android app development was not established because of the complicated and time consuming CI setup. A TaaS solution was introduced as an alternative. For the *MedTabImager* the setup time using TaaS was reduced to a fraction and worked properly. Stability and quality of the TaaS solution have to be further investigated.

References

[Ch14] Chien-Hung Liu; Chien-Yu Lu; Shan-Jen C.; Koan-Yuh C.; Yung-Chia H.; Weng-Ming C.: Capture-Replay Testing for Android Applications. In Computer, Consumer and Control (IS3C), 2014 International Symposium, pp.1129-1132, 10-12 June 2014

[Fe13] Feitelson, D.; Frachtenberg, E.; Beck, K.: Development and deployment at Facebook. In IEEE Internet Computing, vol. 17, no. 4, pp. 8-17, 2013.

[Ga14] Gao, J.; Wei-Tek T.; Paul, R.; Xiaoying B.; Uehara, T.: Mobile Testing-as-a-Service (MTaaS) -- Infrastructures, Issues, Solutions and Needs. In High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium, pp.158-167, 9-11 Jan. 2014

[HF10] Humble, J.; Farley, D.: Continuous Delivery. Addison Wesley, Boston, MA, 2010.

[Kl15] Klepper, S.; Krusche, S.; Peters, S.; Bruegge, B.; Alperowitz, L.: Introducing Continuous Delivery of Mobile Apps in a Corporate Environment: A Case Study. In Rapid Continuous Software Engineering (RCoSE), 2015 IEEE/ACM 2nd International Workshop, pp.5-11, 23-23 May 2015

[Ko15] Kochhar, P.S.; Thung, F.; Nagappan, N.; Zimmermann, T.; Lo, D.: Understanding the Test Automation Culture of App Developers. In Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference, pp.1-10, 13-17 April 2015

[KK13] Kirubakaran, B.; Karthikeyani, V.: Mobile application testing — Challenges and solution approach through automation. In Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference, pp.79-84, 21-22 Feb. 2013