19

September 2012

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Mobile App Testing

Daniel Knott

Best Practices in Mobile App Testing

When testers think about software testing, they will most likely check the documentation, functionality, API, performance, and make sure the software is secure, along with the many other things that depend on that particular piece of software. When it comes to mobile testing, testers have to think about mobile-related functions based on the user's mobile usage patterns.

This article is based on my work experience as a software quality assurance manager in an agile software development team, focusing on mobile apps for iPhone, Android, Windows Phone 7, and mobile web apps. During my daily work within the XING mobile team and exchanges with other mobile test experts, I have gained a lot of insights into the challenging job of mobile testing. Over time, I have defined mobile best practices that helped my colleagues and I to improve test activities and to provide our customers with a higher quality app. Some of these best practices will be covered in this article.

Functional Testing

Every new feature that is developed needs to be tested. Functional testing is an important aspect when it comes to mobile app testing. Based on the developed test cases, mobile testers should do manual and automated testing. At the beginning of the test phase, a tester must test the mobile app manually as a "black box" to see if the functions provided are correct and work as designed.

Besides textbook software testing, like clicking a button to see what happens, mobile testers must perform more functional, mobile-device-specific testing. Today, modern mobile devices have a touchscreen requiring multi-touch gesturing to interact with them. Devices can be used in portrait or landscape mode. They provide motion, tilt and gyroscope sensors. They have different interfaces to communicate with other devices or services like GPS, NFC, cameras, LEDs and many more.

A mobile software tester must be sure that the app will work with all these specific device functions if they are used within the app. The sheer number of different mobile devices means that it is not possible to cover all of them during testing, so testers need to focus on key areas of their app during functional testing.

What has proven to be very useful, yet simple, is device rotation. During my testing activities, I found so many bugs just by rotating the device from portrait to landscape mode and back again.

Besides the entire manual testing process, it is really important to have good test automation in place for mobile applications. Every code change or new feature could affect existing features and their behavior. Usually there is not enough time for manual regression testing, so testers have to find a tool to perform automated regression testing.

Currently, there are a lot of mobile test automation tools on the market, both commercial and open source, for each different platform like Android, iPhone, Windows Phone 7, BlackBerry, and mobile web apps. Depending on the development strategy and infrastructure, quality assurance experts need to find the test automation tool that best fits their environment.

From an Android perspective, there are open source tools like Robotium [ROB01], Robolectric [ROB02], Roboguice [ROB03], MonkeyTalk [MON01], Monkeyrunner [MON02], NativeDriver [NAT01] and Calabash for Android [CAL01]. The Android test automation tool Robotium has become the de facto standard in the open source world, as it is able to simulate real user interaction on real devices. It is really easy to use and is based on Android test instrumentation.

iPhone test automation tools include KIF (Keep It Functional) [KIF01], UIAutomation [UIA01], MonkeyTalk [MON01], Calabash for iOS [CAL02], Frank [FRA01], Zucchini [Zuc01], and many more. All of these tools are also able to simulate real user interaction on a device or on the iOS simulator

Choosing a tool for test automation is not easy, but one fact should always be kept in mind when making a decision, as it is crucial – the test automation tool should use the same programming language as the production code. If the test and production code are written in the same language, it provides a number of benefits for both testers and developers, as it makes it easy for them to do pair programming. Testers can exchange with developers on the same level, they can ask questions related to the programming language, and they can perform code reviews of the test and production code. When it comes to test automation, developers can write their own scripts in the language they are used to.

Summary:

- Test the app as a "black box" and try to break it.
- Open every screen of the mobile app and change the device from portrait to landscape mode and back again.
- Don't forget to test device-specific functions, like sensors and communication interfaces.
- Write test automation scripts for mobile apps.
- Choose a test automation tool that fits into the company strategy and infrastructure.
- The test and production code should be in the same language.

Non-Functional Testing

Another important area of mobile app testing is testing the non-functional requirements of a mobile app. There are several issues that mobile testers need to test before a release or further development.

The first tests to be performed during the early development phase should be usability tests. These should be carried out by alpha users or work colleagues. Go to a café or restaurant and ask people questions about their app usage. Show them a first version of the current development state and collect their feedback. See if the users are able to work with the new features to get a first impression.

Inspect the performance of the app. Compare the released version with the current version to see if the performance is the same, better or perhaps even worse. Install the app on older devices to see if the app still runs on them, despite poor hardware specifications. Do the same with state-of-the-art devices as well.

Test how the app reacts to incoming phone calls, SMS, MMS, tweets or other notifications. Check the battery drain while using the app. Be sure that the test device is fully charged for testing and verify the battery usage every 10 minutes to see if the app is consuming too much power. Install the app on devices with a very low battery level and check what happens.

Validate the app's memory usage. If the app is storing data on the local file system, test the usage of different memory cards. Consider what will happen when the local storage is nearly full – will the app crash or notify the user with a proper error message?

Test the app's installation and deletion process. Even more importantly, test the update process from an older to a newer version. Maybe the local database has changed, which in turn could cause some serious migration problems.

Is the app localized? Testers need to test the app in different languages. Make sure that every string has been translated into the required language.

Don't forget to test on different network carriers and at different network speeds. Make sure the app works with GPRS, EDGE, UMTS, LTE and WiFi. Don't forget to check how the app reacts if the network connection is sketchy or drops completely. Use the app in flight mode and see what happens if a request fails.

Connect the test device to a computer and validate the development log files to see if there are any exceptions, warnings or other strange abnormalities there.

These are just a few of the non-functional requirements mobile testers and developers should think about when developing and testing an app. It is simply not possible to check everything, and the whole team should support QA members in covering most of the above scenarios in order to prevent the user from receiving a bad experience.

Summary:

- Do usability testing.
- Compare performance levels between the released and the new version of the app.
- Check how the app reacts to incoming calls, SMS, MMS, or tweets.
- Validate the battery drain of the test device.
- Test the app's memory usage.
- Install and delete the app.
- Test updating from an old to the latest version.
- Verify the translation into other languages.
- Use the app on different carriers and network connections like GPRS, WiFi, or LTE.
- Check the log files for errors or exceptions.

Test Devices – Fragmentation

The key question regarding mobile test devices for a mobile quality assurance person is, *"Which devices are right for testing?"* This question needs to be answered because it is simply not possible to do testing on every device!

As things stand, there are two big players on the mobile device market: Android and iOS! But there are several other platforms that are used fairly often, depending on the geographical region. These are Windows, BlackBerry, webOS, SymbianOS, and feature phones. The chart (*Figure 1*) shows smartphone operating system usage by vendor in China, Germany, and the USA.

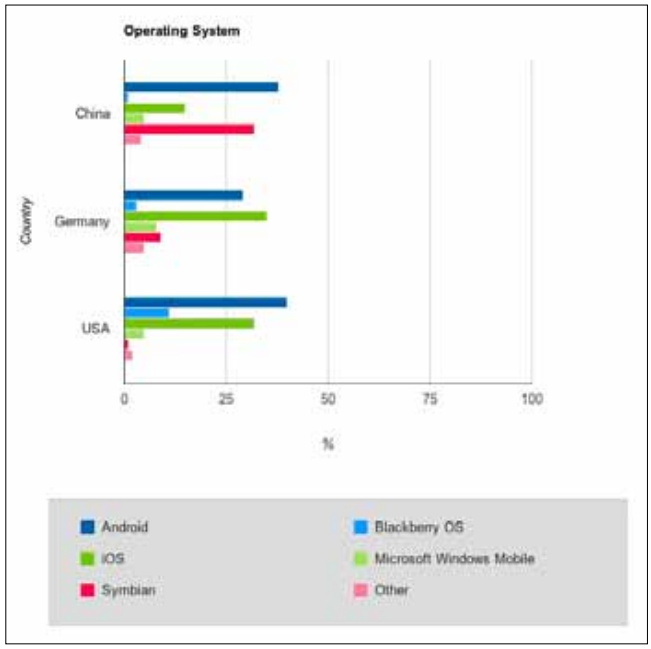

Figure 1: Smartphone operating systems by vendor. Data from www.thinkwithgoogle.com/mobileplanet/en

Nearly every platform has different device vendors who sell smartphones with different hardware and software specifications, as well as customized user interfaces. For example, in the Android world there are vendors like Samsung, HTC, ASUS, LG, Motorola, Sony, Huawei, and many more. This is a serious case of device fragmentation and it is really difficult to find the right devices for testing.

Thinking about mobile websites is another challenge that can be really painful, due to the various mobile browsers such as Safari, Opera Mini, Dolphin, Android and RIM native, Google Chrome, Firefox, Internet Explorer 9, and some other feature phone browsers!

So what is the right choice for testing? Just use the latest devices and browser versions? Buy every device on the market? Or use simulators or emulators?

A little side note about simulators and emulators: don't use them for testing! They may be useful for basic testing, but the results are not the same as on real devices.

In my opinion, a really good approach to solving the test device problem is to group the devices and browsers. For example, mobile testers can group devices depending on their hardware and software specifications. Each group is assigned a priority, like A = highest, B = average, and C = lowest. Each group contains devices that are assigned to that category, depending on the platform and vendor.

Overview of possible group categories:

- **Group 1, priority C:** Small devices with a small CPU, RAM and low resolution. Older software versions and older browsers.
- **Group 2, priority B:** Mid-range devices with an avg. CPU, RAM (<512 MB), good screen size and resolution. The software is not the latest.
- **Group 3, priority A:** High-end devices with a dual/quad-core CPU, RAM (>512 MB) and a high screen resolution. Latest software versions.

These three groups cover most of the users on a specific platform and also represent other phones on the market that fit into the same group. This will downsize the amount of effort required during developing and testing.

Summary:

- Group and prioritize test devices and browser versions.
- Do not use simulators and emulators for testing.

Combine Tools

As mentioned before, mobile testers must do test automation for mobile apps in order to make sure that code changes do not affect current functionality. Another best practice is to combine testing tools and integrate them into a continuous integration server in order to execute them from a central place. Developers need to write unit tests for their code to be sure that each small component is safe and works as expected. In addition, it is very useful to use tools like Robotium or Keep It Functional to implement end-to-end integration tests, which behave in the same way as a user.

Summary:

- Combine testing tools and integrate them into a continuous integration system.

Internal Beta Releases

If a mobile team wants to get early beta testers of the mobile app, they can set up their own internal app store, e.g. for Android and iPhone. With the hockeykit [HOC01] tool, the team is able to distribute new versions of the app to colleagues via the company WiFi. This is an extremely helpful way of getting initial feedback from colleagues, especially if the team or tester doesn't have an opportunity to show the app to the outside world. Hockeykit also provides useful statistics about how well the app has been tested and which OS versions and devices are used by colleagues. It also includes a crash reporter to see what causes errors and crashes in the current development version.

Summary:

- Use internal beta releases to get early feedback.

Know the Customer

Mobile testing is far more effective if the developer and testing team knows the customers who will use the app later on. A good place to gather information about customers is the app store of the specific vendor (Apple App Store, Google Play Store, Windows Marketplace, and so on).

If a team already has an app in one of the stores, they will receive information about devices, software versions, languages, and carriers used by the customers. Figure 2 below is an example from the Google Play Store which shows statistics about a released Android app.

When it comes to mobile web pages, there is no app store to provide such user data. It therefore makes sense to collect information about the 'user_agents' (devices, browsers) used within the mobile website. With that knowledge, the team can optimize and decrease the amount of development and testing effort required for the various devices and software versions.

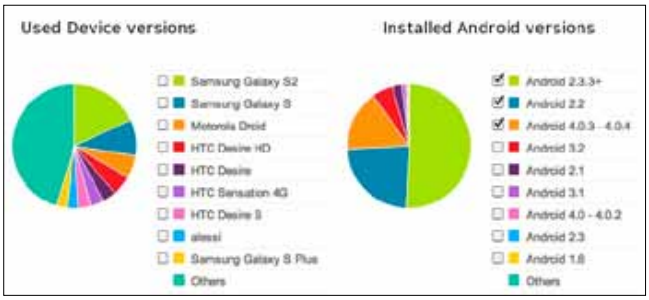

Figure 2: *Statistics from Google Play Store about used Android devices and OS versions.*

Besides user statistics, customer reviews in the app store should be carefully digested in order to collect vital feedback and feature wishes, and react to reported bugs.

Summary:

- Know the devices and software versions your customers use.
- Use statistics from vendor market places.
- Take app store reviews seriously.

Be Active in Communities

This is one last best practice that is not directly relevant to daily mobile testing activities, but may help you to think differently and get new ideas. Be an active software tester in communities like:

- utest (*www.utest.com*)
- mobileqazone (*www.mobileqazone.com*)
- softwaretestingclub (*www.softwaretestingclub.com*)
- etc.

In such communities, software testers can exchange views with other (mobile) quality assurance experts on various testing topics. They can share their knowledge and help other people to solve problems they encounter. Testers can receive great new ideas when it comes to writing test cases, on how to write good bug reports, and improve their own testing and test automation skills. ■

> about the author

Daniel Knott has a technical background involving various different programming languages and software quality assurance tools. He has been working in the field of software development and testing for seven years, and has been working at XING AG in Hamburg, Germany, since 2010. During the course of several projects, e.g. XING search and XING recommendations, he was responsible for test management, test automation and test execution. Daniel's currently position is Team Lead Quality Assurance for the XING mobile and XING API teams. Within the XING mobile team, he is also responsible for the test management and test automation of the XING Android and iPhone apps. Daniel has a broad range of experience in software test automation involving tools like Robotium, KIF (Keep It Functional), Selenium and Java. He has also held presentations at various agile conferences and regularly posts on his blog (*www.adventuresinqa.com*) and the XING devblog (*devblog.xing.com*).

XING profile: *www.xing.com/profile/Daniel_Knott*
Twitter: *@dnlkntt*