WHITEPAPER

sqs.com

Open Source Mobile Automation

Authors: Eric Murphy
Automation Engineer

Justin O'Mahony
Automation Engineer

Niall Conroy
Senior Automation Engineer

Roger Ashwood
Head of Automation Service

SQS Software Quality Systems Ireland Ltd.

Published: August 2015

ERIC MURPHY

Automation Engineer
eric.murphy@sqs.com

Eric joined SQS, having previously worked in the payment card industry. Following initial work on Web Automation projects, Eric now specializes in Mobile Automation and specifically delivering innovative Open Source solutions using Appium. He is a key member of the Mobile and Automation Working Groups within SQS Ireland.

JUSTIN O'MAHONY

Automation Engineer
justin.omahony@sqs.com

Justin is an experienced software tester currently specialising in Test Automation. He is a key member of the SQS Ireland Automation Working Group which involves research and evaluation of both commercial and open-source automation solutions. Justin is currently working on a Mobile Automation project for a major European airline.

NIALL CONROY

Senior Automation Engineer
niall.conroy@sqs.com

Niall is a Senior Automation Engineer at SQS. He is responsible for research and development of the Open Source Web Automation framework and its evolution toward a multi-platform solution incorporating Mobile components. Niall is a member of the SQS Ireland Automation Working Group and is responsible for upskilling SQS Automation resources in the latest tools/techniques.

ROGER ASHWOOD

Head of Automation Service, Ireland
roger.ashwood@sqs.com

Roger is an SQS Principal Consultant with 10+ years of Quality Assurance management and testing experience. He is Head of Service for Automation and Mobile testing in Ireland working closely with clients to provide innovative and effective automation solutions. This includes management of all automation activities from tool/framework evaluation through to implementation and review.

Contents

Management summary

Given the increasing pressure on companies to deliver apps to market in short times, the testing of these apps is becoming a huge financial overhead. Updates to mobile operating systems and greater competition between device manufacturers to produce the latest new devices is taking its toll on mobile testing as there is a more frequent need for regression testing across an ever-expanding range of devices.

Many of the mobile automation solutions that exist currently are expensive and generally require tests to be written separately across the platforms being tested. This increases potential duplication and maintenance issues between automation solutions for Android and iOS. Additionally, most tools require the app to be instrumented, involving the injection of specific configurations into an app in order to integrate with the automation tool in use. This is not ideal as the injected configuration must be removed before shipping the app for live release, so testing, in effect, is not executed against the app that is released.

This paper focuses on a mobile automation solution that was delivered for a large client with a requirement to deliver a native mobile app for both the iOS and Android platforms going to market at the same time. The solution implemented was based on open source tools such as Selenium [1] and Appium [2]. It enabled tests to be written that could be run across multiple devices simultaneously, reducing time to market and man hours of manual test effort, and increased testing coverage across devices that would otherwise not be tested due to time constraints. The solution proves that it is possible to build a robust mobile automation solution in which the same set of scripts can be run across both platforms and also in which instrumentation is not required.

Introduction

The increasing trend among companies to move to agile practices enabling faster time to market, has resulted in automation becoming a key part of the testing process when building mobile apps. The relative immaturity of the market and proliferation of devices and platforms results in specific challenges.

This paper reviews the current state of the mobile app market, delves into the many challenges of mobile automation and then looks at how these challenges were addressed by the choice of automation techniques and tools used in the solution presented.

Market – current status and outlook

The mobile app market is massive and continuing to grow. When Apple went live in July 2008, its App Store contained only 500 apps. By the end of 2014, this figure had accelerated to over 1.2 million apps. Similar trends have been observed on Google Play and Amazon AppStore (Figure 1).

Given that the industry is relatively new and that users are shifting to mobile device use over conventional desktop/PCs, the level of growth is set to continue. Google Play, for example, is now delivering apps from over 350,000 different developers [3]. In 2014, all three of the main players – Google, Amazon and Apple – had increased the numbers of apps by ~50% over the year. More and more businesses are seeing mobile as an essential element of their digital strategy. 128,000 new business apps were released in the Apple Store in 2014 [3].

In fact, there is increasing evidence to suggest that companies are more interested in driving traffic toward their native app rather than through their website. Flipkart, India's largest e-commerce platform with over 22 million registered users, recently announced that it is shutting down its mobile website to further enable the shift towards becoming an entirely mobile-app only shopping platform. The reasons cited for preferring app traffic over web traffic include the fact that it is easier to collect and target user-specific data and advertising, and also the user is always logged in, thus making it quicker and easier to complete transactions and further enabling sales [4].

This level of growth brings varying challenges. Traditional development and test practices struggle to keep pace with the required time-to-market demands. The proliferation of device types and platforms (OS and browser) increases the overall testing scope, coverage and complexity.

Mobile automation can be used to introduce efficiencies in this area; however, given its relative immaturity and the limited support from mobile device vendors, it is currently very difficult to implement effectively.

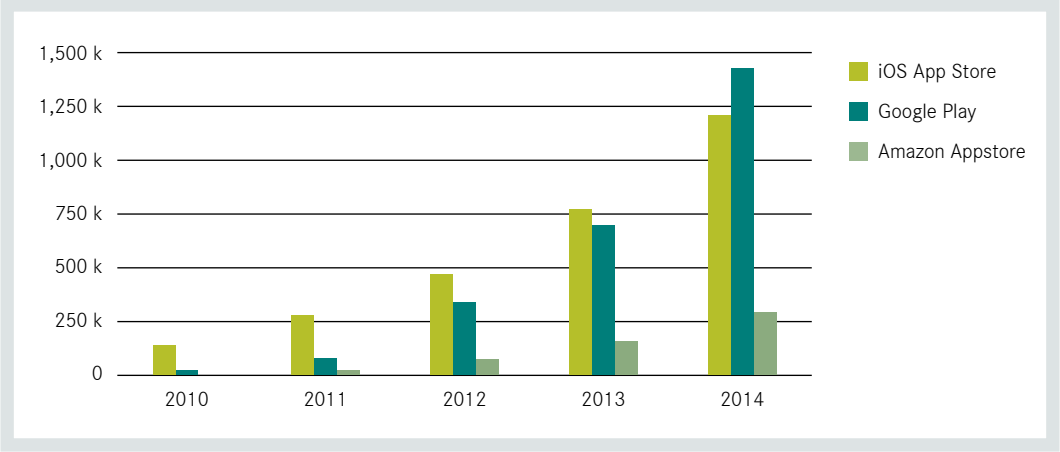

Figure 1: Total number of Apps by App Store [3]

Mobile automation challenges

This section highlights the key challenges/considerations required when automating mobile apps.

Device / OS / browser proliferation

When developing an app, most businesses will need to deliver to customers across the two main platforms; iOS and Android. This typically leads to two development teams, two apps and huge overheads for manual test teams to verify against extensive device/OS combinations. From an automation perspective this results in;

- Different labelling across platforms requiring the use of different object recognition techniques
- Different behaviour (scrolling, drop-downs) between OS versions
- Different behaviour across different devices on the same platform
- Regular OS updates introduce further complexity/risk. An OS update can cause problems with the automation tooling implemented that may require workarounds or patches to resolve.

Environmental requirements

Mobile app automation requires investment (time and money) in hardware and software. Requirements include;

- OS development environments (e.g. Android SDK and XCode)
- Dedicated device hardware (e.g. Mac machines for iOS)
    - For iOS, only one device or simulator can be run on a Mac machine at a time. Careful management and planning are therefore required when scheduling tests across multiple devices at the same time
- Version control software
- Continuous integration software

Automation tool maturity

The current mobile automation tools on the market are in their infancy. Tool selection, whether it is commercial or open source, is critical to the success of any solution.

- Commercial tools usually require the mobile apps to be instrumented before the tool can interact with the app on a device. This involves altering the delivered code to apply the instrumentation configuration. If the App is delivered by a third party vendor, there could be reluctance to provide access to the code base to apply the necessary configuration
- OS updates can cause automation solutions to fail until a fix or workaround is identified and distributed
- Execution performance on physical devices can be poor
- Both Android and Apple provide tools specific to their own platforms which are required for their Apps to be automated. These tools are often unstable and unsupported.

Automation strategy

Given the challenges mentioned and the technical complexity in this area, selecting the appropriate automation strategy is key to the success of any mobile automation implementation. Important decisions need to be made around the following areas;

- Device selection
  - The device selection process for manual and automated testing
  - The use of emulators versus physical devices
  - The use of cloud-based services to test against real devices or building in-house capability
- Development methodology & support
  - Defining the build and delivery process of the apps to the test team
  - Assistance from developers is required to support the set-up of app builds for automation
- Automation framework build & set-up
  - Developing the solution in such a way that it handles cross-platform functionality
  - Version control process of the app and automation solutions
  - Continuous integration and efficient scheduling of test runs
  - Design to allow parallelism, which is vital to reduce the time of test runs
  - Reporting status and triaging issues identified through automation
  - Extendibility to other areas. As well as mobile automation, there may be a requirement to extend the solution to test other areas. For example, as part of a rebranding initiative, an organisation may update mobile and web channels to have the same corporate identity.

SQS open source mobile automation solution

To ensure the delivery of an effective solution, the correct tools need to be chosen to address the problems listed above while also presenting a solution to test across both iOS and Android platforms. In this section we review an implementation of the SQS mobile automation solution successfully delivered for a large European airline who were releasing both an iOS and Android app to the market at the same time. The solution used the SQS open source framework as a starting point and then extended this to suit the specific requirements of the client.

Solution components

The key components of the solution will be detailed in Table 1 along with the reason for their choice.

The solution is comprised entirely of open source tools. The main advantage of open source tools is that they do not require a licence fee. However, there are other advantages:

- Open source tools can be expanded to suit specific project requirements
- Large communities exist which use, develop and support the tools, allowing for efficient debugging of any issues which may occur.

Component	Description
Selenium	Selenium is by far the most popular open source web testing tool available, with a vast number of contributors, both commercially and individually.
Page Factory	• The framework used a Page Factory design pattern. This is a useful feature because discrepancies in the app across both platforms can be handled by the page logic, which allows for the same element to be identified across platforms, even if it has a different locator in each platform. • The diagram on the right side shows how similar objects are set out very differently across iOS and Android. iOS Calendar  Android Calendar 
Appium	• The mobile element of the framework is driven by Appium. • Appium takes Selenium commands and turns them into native mobile actions. • It allows tests to be written across both iOS and Android platforms and does not require instrumentation on the App.
TestNG	Tests are built, maintained and executed via TestNG. In the solution TestNG manages; • Test setup • Environment configuration • Test execution including multi-threading and the load balancing of tests across Selenium Grid
Selenium Grid	• Tests are distributed across different devices using Selenium Grid. • Appium instances connect to the Grid and each instance is then allocated to a dedicated device, emulator or simulator.
Maven	• Maven manages dependencies (such as Selenium and TestNG) and ensures that they can be seamlessly integrated into the solution. • Using Maven, all information about the project to be managed is in a single xml file, the Project Object Model (POM). • Maven also allows for management of test suite runs across different versions of the app and across different devices.
Git	• Git is a version control system that is used for management of codebases. • Effective source control was vitally important as the versions of apps increased during build and testing activities.
Jenkins	• The solution is fully integrated using the Jenkins CI tool. • Builds of the mobile apps were made on a Jenkins server. • Jenkins was also used to build the automation framework itself as well as scheduling test runs.

Component	Description
Dashing & ReportNG	• Reports are created using the ReportNG tool; these are then emailed to the relevant project stakeholders after each run. • Results are also posted to a central dashboard using Dashing, a Ruby-based platform which displays nightly run results, as well as other information about the run.

Table 1: Key components of the solution

The solution developed is designed in such a way that it combined all the above components to allow for a situation where a test can be written once and tested across different devices and platforms.

The app was developed in an agile environment. This ensured that communication with all stakeholders, most importantly developers and testers, was paramount. Developers assisted by ensuring that elements in the app were labelled so that they could be identified by the automation tools. Communication with the test team was vital to decide what tests should and would be automated.

Figure 2 shows the path of an automated test run from end to end and how it integrates with the client's continuous integration process.

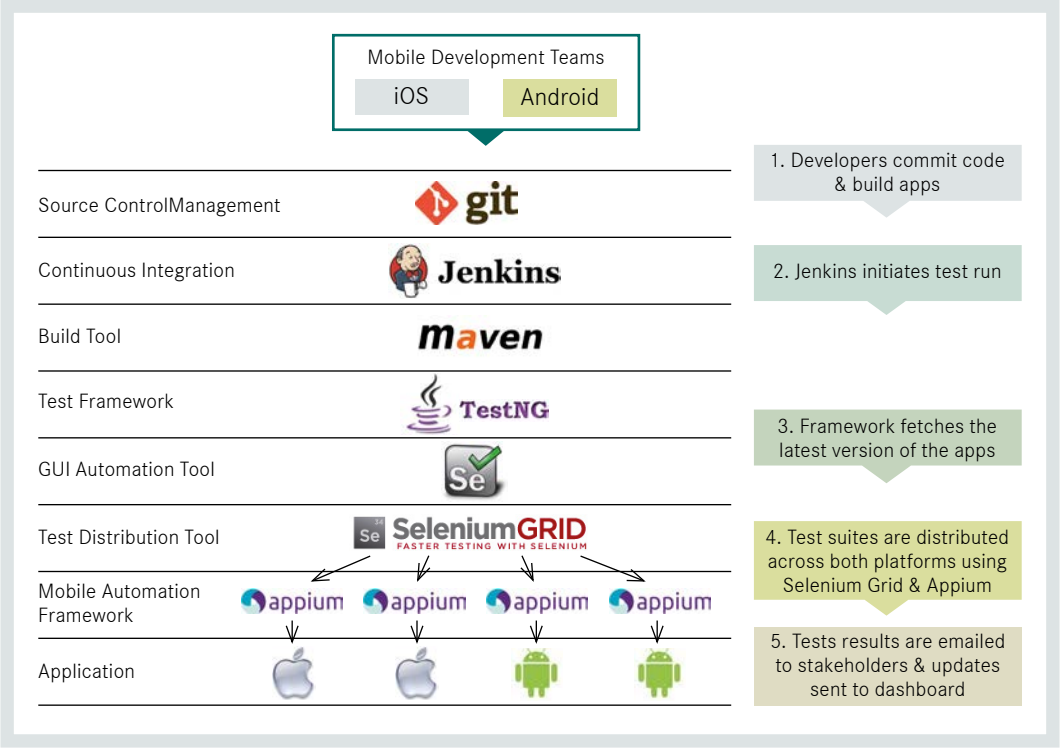

Figure 2: End-to-end automation process

Solution benefits

The benefits of any successful automation solution are numerous:

- Reduced testing time
- Reduced costs
- Increased coverage

The solution implemented above delivers all the expected advantages of automation and more.

- **Continuous integration** – The solution was integrated with the internal CI infrastructure of the client, enabling the testing to become part of the development process. A new version of the apps was built each night. As the code was "checked in" by the development team, an automation regression suite was executed automatically against both Android and iOS versions, with the results distributed to key stakeholders.
- **Versioning** – As one version of the app was released and the new version was under development, automated regression suites could be run against any version of the app at any time. Managing different versions of the automation code became important, as the solution modelled the behaviour of the app at different stages of its development.
- **Instant test runs and results** – The design of the solution was such, that as many tests as hardware would permit could be executed at once. The management of this activity became vital as the number of tests increased. Over 100 tests could be run in under an hour if the hardware was available, providing instant feedback to developers following any code commits. This was particularly useful when working with "hot fixes" that needed to be tested and released as soon as possible.
- **Reporting of app quality** - Test run results were emailed to stakeholders including test leads, business owners and developers. Furthermore, the dashboard allowed the whole team to have visibility not only of the automation results, but information about the sprint as a whole. The quality of the app was instantly visible to all parties.
- **One solution, two platforms** - The solution allowed one central automation team to manage automation across both platforms, thus mitigating the potential risks of having a separate framework for each platform. These risks include:
    - Varying levels of test coverage
    - Two different skillsets required to maintain each solution
    - Duplication of functionality such as reporting and integration with other tools.

Conclusion & outlook

The design and structure of the solution highlighted in this paper is an example that demonstrates how multi-platform automation frameworks are not only possible but can generate real value. The solution fitted the organisation's needs and addressed many of the pitfalls and issues associated with mobile automation. As the app continues to be developed, the importance of a stable automation solution will become more significant. The framework also enabled a level of test coverage on devices which would not be feasible with manual testing alone.

The mobile app market is continuing to expand at an almost exponential rate. There is ever increasing pressure on organisations to reduce time to market. A robust, repeatable and reliable test strategy is required to ensure and maintain high levels of quality of delivered solutions.

Mobile automation should be an integral part of any test strategy to assist test teams. Mobile automation, however, is difficult, immature, and pitfalls do exist. There is a requirement to invest in people and resources to achieve success. Nevertheless, a robust and reliable solution can be delivered that will add real value to the project.

References

[1] http://www.seleniumhq.org/

[2] http://appium.io/

[3] AppFigures (2015). App Stores Growth Accelerates in 2014. http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/

[4] LiveMint (2015). Flipkart moves towards becoming app-only platform. http://www.livemint.com/Industry/J9VeQxowSOIHU8ZMUParUL/Flipkart-moves-towards-becoming-apponly-platform.html

SQS Software Quality Systems AG
Phone: +49 2203 9154-0
Fax: +49 2203 9154-55
info@sqs.com | www.sqs.com