SOLVING THE MOBILE TESTING CONUNDRUM

Even though mobile testing is complex, it can be done successfully with the correct strategy.

A sound mobile test automation strategy must include test automation frameworks, real devices, and emulators and simulators.

SAUCELABS

A global overview by WeAreSocial.com revealed that more than half of the world's web traffic now comes from mobile devices. So, it is not surprising that today's software development is based on a mobile-first or even a mobile-only imperative. But for all these dramatic changes, developers still struggle when it comes to mobile testing. This paper will outline the factors to consider when building an effective mobile test automation strategy and weigh up the benefits and disadvantages of popular test automation frameworks.

There are many factors that can make or break the success of a mobile app. For example, consider device fragmentation. With the advent of new smartphones released every year and a trend of mobile phone users hanging on to their devices for a longer life span, this further complicates the amount of firmware and hardware testing to be completed. When your app needs to work across a variety of devices, it can become a serious burden for testers.

Another factor that contributes to this complexity is the device itself: varying screen and device sizes, resolutions and orientations. Next, we have multiple app types, such as web, hybrid and native. These multitudes of devices operate differently on various device and OS combinations. Finally, you have users all over the world in different regions that must be tested for translations, timezones and targets. These factors make testing with mobile a challenge. The good news is that even though mobile testing is complex, it can be done successfully with the correct strategy. When building a mobile testing strategy, there are three key areas of focus:

1. Real device testing
2. Emulator and simulator testing
3. Test automation frameworks

By focusing on these three areas, organizations will thrive in the fast-paced world of mobile software development.

SCALE CONTINUOUS TESTING WITH EMULATORS AND SIMULATORS

For several years, the use of emulators and simulators to test mobile applications has been met with some resistance. This is based on the perception that if you're not testing on a real device, you're not testing at all.

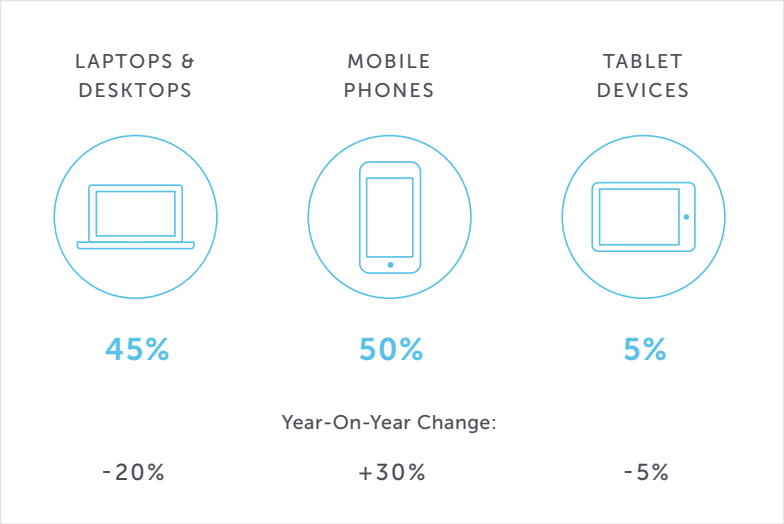

Figure 1: The user numbers driving the mobile challenge all use cases

Although real devices give more accurate test results, using them alone is not ideal for continuous testing and continuous delivery. Due to budget issues, some organizations forgo real devices altogether as they're too expensive, and instead opt for emulators and simulators. But the reality is that effective and efficient mobile app development requires both emulators/simulators and real devices.

An emulator, as the term suggests, emulates the device software and hardware on a desktop PC, or as part of a cloud testing platform. The Android (SDK) emulator is one example.

A simulator, on the other hand, delivers a replica of a phone's user interface, and does not represent its hardware. It does not run the real device OS; rather, it's a partial reimplementation of the operating system written in a high-level language. The iOS simulator for Apple devices is one such example.

Emulators give teams the ability to implement parallel testing and test automation via external frameworks like Appium or Espresso. Selenium revolutionized the world of web app testing by pioneering browser-based test automation. Today, Appium is its counterpart for mobile app testing. Appium uses the same WebDriver API that powers Selenium, and enables automation of native, hybrid, and mobile web apps. This brings huge improvements in the speed of tests for organizations coming from the manual world of testing on real devices. Emulators enable parallel testing in a way that can't be achieved with devices in a lab. Because tests on emulators are software-defined, multiple tests can be run on tens of emulators at the click of a button without

having to manually prepare each emulator for the tests. Further, automation is easier with emulators as the tests can be executed without manual intervention, and be controlled remotely.

DEVOPS, CONTINUOUS TESTING AND CONTINUOUS DELIVERY

The speed of release and change demands that mobile development is agile. Mobile requires that continuous testing (the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks) is a key component to the overall regression testing strategy. In order to properly build out a mobile regression testing strategy, it is imperative that the dev/test teams are well equipped with the following:

- **A comprehensive web and mobile testing platform**
  building a test lab from the ground up is timely and expensive. The best option is to use a cloud-based, device lab solution that provides an extensive choice of real devices, as well as emulators and simulators. This should also include parallel testing so that test execution can be done in a shorter amount of time.

- **Highly scalable, highly available solution**
  developers and testers need to ensure that the infrastructure for mobile testing allows the team to expand coverage as the test suite grows. The goal is to decrease test execution time while providing fast feedback, and to ensure that the team spends less time chasing false failures and more time on actual testing and defect remediation.

- **CI/CD optimization**
  in order for the regression testing efforts to keep up with the fast pace of continuous delivery, the process must have a tight integration with the development workflow and mobile app delivery pipeline. The goal is to eliminate any unnecessary manual steps, and promote an automation-first philosophy throughout the testing process.

Although mobile regression testing can be a challenge, the risks of flaky software can be reduced by making sure your organization has the right strategy in place.

Mobile test automation frameworks

Test automation frameworks are an integral part of mobile test automation. There are many test automation frameworks freely available, but the top mobile testing frameworks are Appium, Espresso and XCUITest. Sauce Labs

recently conducted a survey of teams using test automation frameworks for mobile app testing. The survey found that 63 percent of the participants use the Appium framework, while 31 percent didn't use any test automation framework at all. Since Appium is based on Selenium (the popular open source functional testing framework), it wasn't a huge surprise that most of the users surveyed were using Appium. But the main question is why is it that over 30 percent of users were not using some sort of testing framework? The answer lies in the implementation, or lack of a mobile testing strategy.

When implementing a mobile testing strategy, it is very important to understand

1. The skillset of the test automation team and
2. The framework that best fits the organization's preferred development methodology.

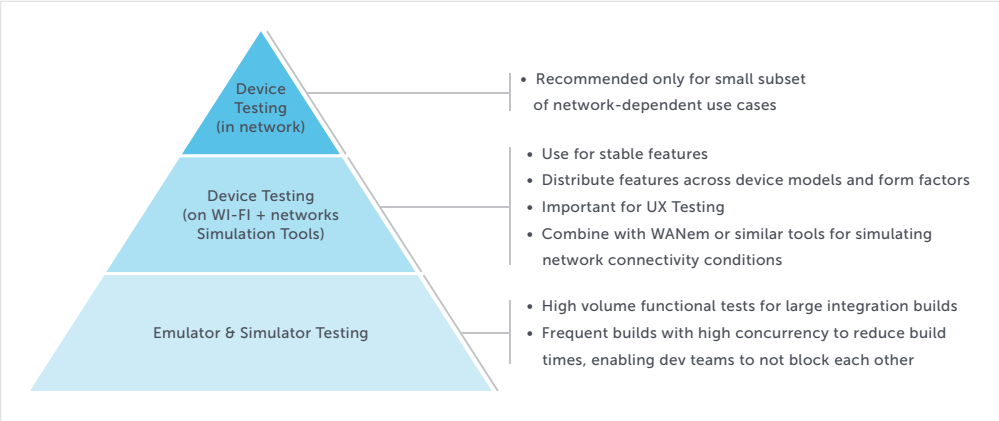

Figure 2: Emulators, stimulators and real devices cover all use cases

TESTER AUTOMATION SKILL SET

When evaluating frameworks, it's important to understand the technical background of your team. Some frameworks take a "black box" test approach, which is typically best for traditional test automation engineers. While other frameworks take a "white box" approach suitable for developers. There are pros and cons to each framework, but let's explore the top three.

Appium

From Appium.io:

"Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS and Android platforms. Native apps are those written using the iOS, Android, or Windows SDKs. Mobile web apps are web apps accessed using a mobile browser (Appium supports Safari on iOS

and Chrome or the built-in 'Browser' app on Android). Hybrid apps have a wrapper around a "webview" – a native control that enables interaction with web content." The key advantages of Appium are:

- A single, cross-platform framework for both Android and iOS
- The ability to automate mobile web, native and hybrid applications
- Tests can be run on emulators, simulators and real devices

Supports any language supported by Selenium, such as Ruby, Java, JavaScript, Python, C#, etc.

Appium is one of the more popular testing frameworks for those organizations that typically have testers who have worked with Selenium and are currently making the transition to mobile development. Since Appium is a black box type framework, the tester has less insight into the code, and the focus of testing is limited to what is exposed by the mobile application under test.

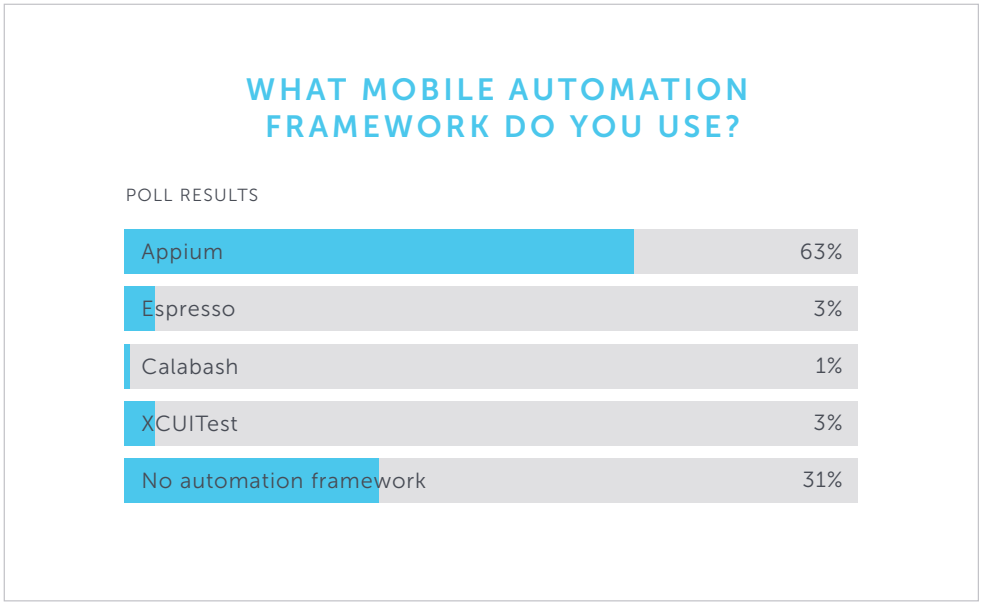

Figure 3: User survey of test automation framework

Espresso

The Espresso testing framework provides a set of APIs to build UI tests to test user flows within an app. These APIs let you write automated UI tests that are concise and that run reliably. Espresso is well-suited for writing white box-style automated tests, where the test code utilizes implementation code details from the app under test.

The key advantages of Espresso are:

Espresso is highly performant and reliable. It is is fully integrated with Android Studio, the preferred mobile development platform for Android apps. It's based on Java or JUnit, which many android developers are familiar with. Since Espresso is integrated with Android Studio, it's automatically CI/CD ready, allowing teams to easily incorporate their test with existing continuous integration & delivery tools and frameworks.

Another advantage of the Espresso test framework is that it automatically provides updates to the latest and greatest features of the Android operating system. This helps keep testing in sync with new features and improvements as they become available. In addition, Espresso's integration with Android Studio provides backward and forward compatibility. This allows teams to ensure their app works on previous releases of the Android OS.

A huge benefit for testing with the Espresso framework is automatic synchronization. Many testers struggle with UI testing reliability because of waits for elements to become visible or active. With other frameworks, the developer must write code to manage these issues. Once the object appears, Espresso handles the execution on that object for you. This results in less "flakiness" of test scripts and more reliable regression test suites.

XCUITest

XCUITest is the automation framework that ships with Apple's XCode. The XCTest framework lets iOS app developers create and run unit tests, performance tests and UI tests for XCode projects. It provides developers with capabilities similar to those found in Espresso. However, Espresso is dedicated to code written for the Android operating system and XCUITest is dedicated to Objective-C and Swift code that runs under iOS.

XCUITest was created specifically for testing iOS apps and is maintained by Apple. This ensures that developers get the best support from the Apple developer community as well as the latest updates for new releases on iOS. Additionally, the ability to run unit, UI and performance tests allows more comprehensive testing within a single framework.

Criteria	Espresso	XCUITest	Appium
Scope	Andriod only	iOS only	iOS and Andriod
Language Support	Java	Swift/Objective C	Java, #C, Ruby, Python
Publishers	Google	Apple	Open Source, Sauce Labs
Testing Artifact	APK and Test APK	Internal Target	APK or IPA
Need access to source to create tests?	Yes	Yes	No
Allows Remote Testing?	Yes	No	Yes
Ease of use for test creation	Easy, has internal recorder	Easy, has internal recorder	Medium, has recorder that produces testing source code

If the development of the app is strictly for iOS then the developer has everything at their fingertips. If the app is cross platform using XCUITest it can be a slight disadvantage - for instance, the issue of developer skill set. Tests written for XCUITest can only be written in Objective-C or Swift code. A team that has developers experienced in Java, JavaScript, or any language that Selenium supports must now learn another scripting language and manage two frameworks. This is another reason Appium has become popular with test automation engineers that support multiple mobile app platforms.

Espresso, XCUITest and Appium are all useful frameworks for writing tests for mobile applications. But a question to be answered is really what is the best framework for the job. No one tool is applicable to all situations. Espresso and XCUITest are great tools for mobile testing when source code is available and the developer is working in an IDE. Appium is the mobile application testing tool to use when all that is available is the API or APP deployment artifacts. And all come with tradeoffs. Finding the right test automation framework for the right job is a hard determination to make.

Testing frameworks are a key component of mobile application development. With Agile, DevOps, continuous testing and integration, developers and test automation engineers must continue to develop, test and execute at a rapid pace.

Frameworks such as Espresso, XCUITest and Appium make automation easier to scale for small to medium sized organizations to large enterprises.

CONCLUSION

A sound mobile test automation strategy must include test automation frameworks, real devices, and emulators and simulators. Each component is critical to the rapid release of realease of high quality software.

Agile, DevOps and continuous testing are the norm. Mobile software development naturally follows this paradigm. Implementing these components as part of the mobile testing strategies will increase the chances of successful releases and decrease the chances of poor app quality.

ABOUT SAUCE LABS

Sauce Labs ensures the world's leading apps and websites work flawlessly on every browser, OS and device. Its award-winning Continuous Testing Cloud provides development and quality teams with instant access to the test coverage, scalability, and analytics they need to rapidly deliver a flawless digital experience. Sauce Labs is a privately held company funded by Toba Capital, Salesforce Ventures, Centerview Capital Technology, IVP, Adams Street Partners and Riverwood Capital. For more information, please visit saucelabs.com.

saucelabs.com/signup/trial

FREE TRIAL

SAUCELABS

SAUCE LABS INC. - HQ
116 New Montgomery Street, 3rd Fl
San Francisco, CA 94105 USA

SAUCE LABS EUROPE GMBH
Stralauer Allee 6
10245 Berlin DE

SAUCE LABS INC. - CANADA
128 West Pender Street, 8th Floor
Vancouver, BC V6B 1R8, Canada

SAUCE LABS - POLAND
Złota 59 St., 4th Fl.
00-120 Warsaw, Poland