WHITE PAPER

Testing Strategies and Tactics for Mobile Applications

Mobile app testing is time consuming and expensive, yet critical to ensuring your consumers have a positive experience when they use your mobile applications. It is vital that you make sure that the experience is a great one for every consumer every time they use your application, starting with the very first time. If you fail to do a good job app testing, this will result in your customer doing it for you—and unlike your testing team, your customers do not have the tools or the time to report back problems. Also, keep in mind that your customers do not want to be treated like guinea pigs. When they find a flaw, you'll never hear a word from them and they certainly won't come back.

Ideally, your developer has done a great job and made no mistakes, but regardless of that, the goal of your testing efforts is not to find errors. Instead, it should be to understand the quality of your app. Does it work? Does it function as expected? Will it meet the needs of your users so they come back repeatedly?

Mobile application testing presents unique challenges. There are tradeoffs that you need to consider and choices that you need to make regarding the mix of different techniques and methods that will be used in mobile app testing. Each testing method you consider will have pros and cons associated with it, and you will likely find that there is not a single testing method that is completely satisfying. Rather, you will need to consider a testing strategy that combines different testing options that as a whole, provide the best overall testing result, balancing the tradeoffs between cost, quality, and time-to-market.

In this document, we examine various testing options for mobile applications while explaining the factors that you need to conside in determining your testing strategy. Finally, we make some recommendations on how you can combine the various testing option to find the best overall strategy to fit your mobile application(s).

Native Applications

For many people, "mobile apps" have become synonymous with native applications (and hybrid applications). Commonly downloaded from an app store, they offer the user a unique experience that maximizes the capabilities of the device and operating systems for which they are developed. The app download is often controlled by the gate-keeping app store, with mechanisms in place to charge potential consumers. This simple and proven monetization model has fueled native app popularity within the development community. Beyond their acceptance in the consumer market, they also allow enterprises to deliver productivity tool to an increasingly mobile workforce.

While native applications can provide a rich experience to the user—and possibly a lucrative one for the developer—they also add some complexity to the lives of those tasked with testing them.

Testing needs to determine whether the app can be successfully downloaded to the device, executed on the device, and interact with the supporting back-end content infrastructure. When updates are made, you need to be sure that the application can be pushed out to and accepted by the end user. There is a misperception that successful testing of app functionality on one device provides assurance across all others of the same operating system.

Native applications are inherently tied to the hardware and operating systems for which they are written. To meet the challenge of testing for native mobile applications, it is essential to test on the physical devices supported by your application. You'll also want t ensure backward compatibility with each older generation of the device you're expected to support. And you'll want to account for updates to the operating systems, especially popular "major" releases that users will apply quickly upon availability.

Web Applications

Like the web itself, a mobile web application is viewable by users around the world. Even if you're initially targeting only users in a single country or on a single network, it helps to understand the global dynamic. While taking a standardized web application approach reduces the need to develop multiple code lines based on operating system, the challenges below still apply.

When testing both native and mobile web applications, there are several challenges. As the nature of each challenge is understood an exploration to manage issues and mitigate risk is done on different technology options. Coming up with the right solutions

determining the technology that best suits your app testing requirements.

Devices: The Biggest Mobile Testing Challenge

The mobile devices used by consumers create the most obvious challenge to mobile testing. Potentially tens of thousands of different client devices could be used to access your mobile app or website, and they must therefore all be considered when testin your mobile applications. Add to this the different versions of operating systems, and the permutations get crazy-big! You can sacrifice coverage across device/OS combinations to an extent, but when you reduce the number of device types that you test against, you are taking a chance that your application might not work for a number of potential customers. To handle the device challenge, you have three options: You can test exclusively using real devices, you can test exclusively with emulated devices, or yo can use a combination of both.

Real devices have the advantage of having all of the limitations and quirks present in the actual client operating system, hardware, and firmware combination used by your target consumers. However, testing with real devices can be expensive, depending on hov you approach it. They are expensive to buy—and forget about the advertised prices, for those are the operator-subsidized prices that come only with a contract that has its own cost implications. You might be able to get a manufacturer or network operator to loan you devices for testing, but you need to join a waiting list and convince the hundreds of manufacturers and hundreds of mobile network operators that you should be their priority. Airtime and subscription costs also need to be paid. And finally, testing with real devices can be disorganized and labor intensive if the testing environment is not conducive to creating, collecting, and reproducing results in a consistent manner.

Emulated devices, on the other hand, are relatively easier to manage. You can switch device types by simply loading a new device profile, and instantly you have a new device that presents itself to your native mobile or web application in the same way that the real device would. And because the emulators run on more powerful PCs and servers and were designed with testing in mind, they are typically fully instrumented to capture detailed diagnostics about the protocols that go back and forth between client and server at the various levels of the stack.

When you encounter an application fault, you will have the information to isolate and thereby correct the problem. Solutions that emulate mobile devices are cost effective because a single platform with frequent updates of device profiles can be used to test every device on the market, now and in the future.

The big disadvantage of emulated devices is that they lack the quirks, faults, and characteristics that only a real device can provide. An emulated device may not give the pixel-perfect accurate rendering that you're assured to have with a real device solution. And while the processing power of your local PC can be an attribute, it will also hide any issues that you may have with the responsiveness of your application. Finally, an emulated device is not sensitive to the ambient conditions that can impact the behavior of the device. In the majority of cases this is a good thing, however if you want to know how well a device performs in an exact location such as a crowded stadium, a real device is your better bet.

Fortunately you're not limited to an either/or selection when determining the right device solution for your mobile app testing needs. A third approach is to select a mix of both emulation and real device testing. First, start testing in an emulated environment to take advantage of the speed and device diversity that an emulator can provide. Emulated device testing early in the developmer cycle can help you achieve these goals at a relatively low cost. Early in the development cycle you don't need the pixel-perfect rendering afforded by an actual device. The risk of not having the nth degree of certitude is easily outweighed by the benefits gained by increasing the number of test cases and device types covered in the test suite. The addition of real devices to your test plan later in the development cycle is helpful for validating that applications are functioning as expected, and certifying that all development requirements and objectives have been met.

Network: A Regional Challenge

There are well over 400 mobile network operators in the world.

Each mobile operator may support multiple network technologies including LTE, CDMA, GSM, and some use less common or local networking standards such as iDEN, FOMA, and TD-SCDMA. Each network has a unique combination of network infrastructure that tunnels the packet-based protocols used by mobile networks into TCP-IP protocols used by the mobile web. Each network operato has implemented systems that behave slightly differently from vendor to vendor in order to perform the required tunneling. Lastly most network operators have inserted mobile web proxies (that is, the gateway) to dictate how, when, and if you are able to connect to a particular site. When a network operator implements a mobile web proxy, it can restrict the flow of information that travels between your server and the test client. Some proxies limit the sites that can be accessed via a phone to only those approved by the operator in what is often referred to as a "walled garden." Other proxies might use "transcoding" in an attempt to scale down fixed web content to better fit onto mobile phones. As you can see, the network challenge is quite complex.

It is not possible to discuss the network challenge without discussing location. It is a simple fact that to fully test the full network stack on a particular operator's network infrastructure, you must be connected to the target network. But the challenge is made more difficult by the fact that the radio signals are not strong on cellular networks, so you must be adjacent to a cell connected to the operator's core network to run your test. For example, if you want to test against the carrier SFR, you must be in France, and if you want to test against China Mobile, you must be in China.

Obviously, traveling to every network operator that you need to support can be very expensive, and there are obvious cost

Network Bypass

When you bypass the network's lower layers, you use TCP/IP to connect directly to the server and you ignore the GPRS tunneling systems used by network operators. Since most real devices are not capable of doing this, you will need to use a device emulator t perform the bypass. Not all device emulators support this feature, so you may want to look for a device emulator that can perform network bypass via the Internet. Some device emulators also have the ability to access the operator's proxy (but only if it is expose to the Internet) to allow a more realistic test. Although the operator's web proxy is available to only its customers, there are test proxies on the Internet that can be used. Even if you don't have a test proxy, you will still be able to test directly against your origin web server.

An advantage of bypassing the network is that you will not have to use or pay for airtime. And because you are using a device emulator, you will benefit again from having a fully instrumented stack.

But often a network bypass cannot emulate the effects and timing of the network and the various network elements such as proxies. Finally, when you use this technique, you can't use real devices and therefore will not be able to see the quirks and limitations that real consumers will see.

Real Networks

Of course, it is possible to test against real networks. One method is to use real devices at the target location, although you will fac many of the problems already discussed. Alternatively, many device emulators support modems that allow you to use your emulated devices on the local network—but again, there is the cost of traveling into range of the network. But there is another option.

One piece of useful test equipment is a real device in the cloud. This type of testing solution consists of a physical handset mounte in a remote box with a remote control unit and a remote antenna. The remote control unit is physically connected to the device's screen and keypad control circuits and is capable of pressing keys and collecting screen images. Exposed to the Internet, this solution lets a user on a local PC or web client control a device with their mouse and keyboard, thereby seeing what is happening remotely on the screen. These devices provide an elegant solution that can be connected to live networks.

Remote real device solutions often have the ability to record a test for subsequent replay, a capability that can be useful for regression testing and automated build acceptance.

Because there are so many different makes and models of devices, it is often too expensive to buy a remote real device solution fo all the devices that you need to test against. Fortunately, you can "rent" testing time on a resource that is shared with others and managed for you. You simply need to open an account, and then buy testing time with a given make and model of device when an where you need it.

Scripting: The Repeatability Challenge

The last challenge of mobile testing is what we call scripting, which is the method of defining a test. Script execution can either be manual or automated. You either write down the scripts in a document or a spreadsheet, which is then used by a test engineer wh manually interacts with the test environment to determine pass/fail, or you run automated scripts that in turn drive interaction wit the device and application, which record the results.

Because there are so many different devices with different interface options, automated scripting needs to be abstracted away from the device to be of any real use. Consider a script that follows strict keystrokes/gestures on an Apple iPhone. This script woul not have any chance of working on a Samsung device, because the user interfaces are different. Fortunately, most real device automated testing software provides high-level scripting that operate on the text, image, or object layer. Most device emulators are capable of automating test execution using a higher-level, abstracted scripting language that is not device dependent.

When you use automated scripting, the cost of setting up the script will typically be higher than the cost of a single manual execution of a test. But if it is a test script that you run on a periodic basis, every time that you subsequently run the script, the more time and effort you will save. If you run the script enough, you will eventually recover the costs of initial scripting.

Finally, many automated scripting tools have a special ability to "spider" or "crawl" a mobile website or application. This is a special capability that can test an entire site with a single command. Although this capability will not be able to perform complex transactions, it is a quick, powerful, and cost-effective tool to set up that will walk through your site or app looking at every page/feature for errors and device inconsistencies.

Recommendations

Hopefully, you now understand a lot more about the challenges associated with mobile testing of native and web applications. But what do you do with this information? What should be your testing strategy for mobile application testing?

It is not a matter of choosing one tool or technique because there are simply too many compromises that must be made. Most likely you will need to use a combination of testing tools and techniques to meet your quality requirements. Generally, you can

1. **Take advantage of a device emulator**. Emulated devices provide a cost-effective means of performing the bulk of your testing in a well-instrumented test environment, quickly and efficiently. You will want to use your device emulator with variou options such as bypassing the network, using the live network via modems, and a good scripting language, so you should lool for these features during your selection process. When you look at device emulators for testing, make sure that they have the instrumentation and the network options to provide you with the flexibility that you will need. Also find out if the emulator ha the necessary diagnostics to isolate problems and the flexibility in network stacks needed to test different network options. Determine if your emulated device solution contains a high-level scripting solution to allow you to replay your test cases over and over again. Finally, look for an emulated device which will allow you to change device profiles quickly.
2. **Invest in a real device cloud**. You might be in a situation where you have to test on a remote live network and not have the necessary device. Having an account with a vendor that lets you access remote real devices at any time is very useful. It is a great solution to have in your bag of tricks.
3. **Automate wherever possible**. Emulators and remote, real-device solutions that support script and playback functionality ar timesavers that enable execution of more test cases with a higher degree of consistency. Clearly, a solution that integrates real and emulated devices is ideal.

Start a free 7-day trial >

Learn more about mobile application testing >