



Testing Strategies and Tactics for Mobile Applications

Testing. No one really wants to do it. It's expensive. It's time consuming. But unfortunately, it's needed to ensure that your consumers have a positive experience when they use your mobile applications. And it's vital that you make sure that the experience is a great one for every consumer every time they use your application, starting with that very first time. Fail to do a good job of testing and your customer will end up doing it for you—and unlike your testing team, your customers don't have the tools or the time to report back problems. And keep in mind that your customers don't want to be treated like guinea pigs. When they find a fault, they simply never come back, and you'll never hear a word from them.

The goal of your testing efforts is not to find errors. Perhaps your developer has actually done a great job and did not make any mistakes. Instead, your goal in testing should be to understand the quality of your offering. Does it work? Does it function as you planned? Will it meet the needs of your consumers, so that they are delighted a come back again and again?

But when it comes to testing mobile applications there are unique challenges. The challenges of mobile testing will present you with tradeoffs that you need to consider and choices that you need to make about the mix of different techniques and methods that you will use in testing. Each testing option you consider will have pros and cons associated with it, and you will probably find that no one testing option will be completely satisfying. Rather, you will need to consider a testing strategy that combines different testing options that together provide you with the best overall testing result that balances the tradeoff between cost, quality, and time-to-market.

In this document, we examine the various testing options for mobile applications while explaining the factors that you will need to consider in determining your testing strategy. Finally, we make some recommendations on how you can combine the various testing options to find the testing strategy that fits your mobile application.

Mobile Testing Challenges

Like the Web itself, your mobile Web application is viewable by consumers around the world. Even if you're initially targeting only customers in a single country or on a single network, it helps to understand the global problem.

When we test mobile Web applications we encounter several challenges presented by the nature of the global, mobile Web. To make a wise decision about our testing options, we must first explore these challenges. As we understand them, we can explore different technology options to manage the challenges in different ways. As you will see, each option has pros and cons that you will need to examine before choosing the combination of testing options and technology that best suits your testing requirements. These mobile testing challenges include *devices*, *network*, and *scripting*.

Devices: The Biggest Mobile Testing Challenge

The mobile devices used by consumers create the most obvious challenge to mobile Web testing. There are tens of thousands of different client devices that could be used on your mobile site, and they must therefore all be considered when testing your mobile applications. This number can be reduced to an extent, but each time you reduce the number of device types that you test against, you are taking a chance that your application might not work on a device, locking out a number of potential customers. To handle the device challenge, you have two options: You can test using real devices, or you can test using emulated devices.

Real devices have the advantage of having all of the limitations and quirks present in the actual client hardware and firmware combination in the hands of your target consumers. However, testing with real devices is incredibly expensive. First, 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 a priority. Airtime and subscription costs also need to be paid. And finally, testing with real devices is labor intensive—and you know the cost of labor.

But there is another aspect of testing with real devices that you need to consider. Real handsets are not designed with testing in mind. The limited processing power and storage of the handsets does not allow on-board diagnostic software to be loaded, so they lack instrumentation. With real devices you will not be able to record the protocols going back and forth between the device and your application, and this will limit your ability to isolate problems and make corrections.

On the other hand, emulated devices are 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 mobile 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 thus correct the problem. Emulated devices are thus cost effective, because a single platform with frequent updates of device profiles can be used to test every device on the market both today and tomorrow. The big disadvantage of emulated devices is that they lack the quirks and faults that only the real device can provide.

Network: A Regional Challenge

There are well over 400 mobile network operators in the world. Some are CDMA, some are 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 operator has implemented systems that behave slightly differently from different vendors 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, thus expanding the number of mobile sites that can be seen—and unfortunately they might also “transcode” your made-for-mobile application. Finally, some proxies strip vital information from the HTTP headers that your application might depend on to provide functionality or to provide device adaptation. As you can see, the network challenge has a lot of complications to it.

It’s not possible to discuss the network challenge without discussing location. It’s 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. Thus, if you want to test against SFR, you must be in France, and if you want to test against China Mobile, you must be in China.

Obviously, travelling to every network operator that you need to support can be very expensive, and there are obvious cost tradeoffs to be considered.

There are different ways of dealing with network challenge it. We can bypass the lower layers of the network and simply test over the Internet or LAN, or we can use the real network by using either phones or modems.

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 to perform the bypass. Not all device emulators support this feature, and you may want to look for a device emulator that can perform network bypass by using the Internet. Some device emulators also have the ability to access the operator's proxy (but only if it is exposed to the Internet) to allow a more realistic test. Even if 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 need to use and thus pay for airtime. And because you are using a device emulator, you again benefit from having a fully instrumented stack.

The disadvantages of network bypass are that is that we often 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 thus don't see the quirks and limitations that real consumers will see.

Real Networks

Naturally, it is possible to test against real networks. One method is to use real devices at the target location, though you will face 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 remote real device. This type of testing solution consists of a physical handset mounted 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 press buttons and see what is happening on the remote device. These devices provide an elegant solution that can be connected to either live networks or simulated networks, although most rely on 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.

Remote real device solutions reduce the cost of travel to foreign networks, but still can be expensive because the cost of the device is now amplified by the cost of the remote control hardware, remote control software, and local software. Because there are so many different makes and models of devices, it is often too expensive to buy a remote real device solution for all the devices that you need to test against. Fortunately, most of the companies that make this type of equipment offer the ability to “rent” testing time on a resource that is shared with others and is managed for you. You simply need to open an account, and you then buy testing time with a given make and model of device when and where you need it.

But these solutions are not without their own downsides. First, the testing will take time to set up and has the same labor cost as testing with a real device. Second, like a real device, the solution lacks the diagnostic tools to capture the information needed to isolate and find corrections to problems with your application. And finally, you will need to make sure that the solution is located where you need it.

Scripting: The Repeatability Challenge

Our last challenge of mobile testing is what we call scripting, the method that is used to actually execute the test script. 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 who manually enters keystrokes, or you run automated scripts that in turn evoke the keystrokes and record the results. Clearly real devices require manual scripting while emulated devices can use automated scripting.

Because there are so many different devices with different menu structures and keystroke options, automated scripting needs to be abstracted away from the device to be of any real use. Consider a script that follows strict keystrokes on an Apple iPhone. This script would not have any chance of working on a Nokia N70, because the user interfaces are completely different. Fortunately, most automated testing software provides high-level scripting functions such as “goto URL” or “send SMS”, which are not dependent on the particular menu structure of the target device. 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 script. 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 cost of initial scripting.

Finally, many automated scripting tools have a special ability to “spider” or “crawl” a mobile Web site. 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 test to set up that will walk your mobile Web site looking at every page for errors and device inconsistencies and is a very powerful and cost-effective tool.

Your Mobile Testing Strategy

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

First, it is not a matter of choosing one tool or technique; 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. But generally you can narrow your choices down based on the following recommendations:

1. *Do invest in a device emulator.* This will allow you to perform the bulk of your testing in a well-instrumented test environment that is cost effective. You will want to use your device emulator with various options such as bypassing the network, using the live network via modems, and a good scripting language, so you should look for these features during your selection process. Finally, 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.
2. *Take advantage of remote real devices.* Having an account with a vendor that lets you access remote real devices at any time is very handy. You never know when you might need to test on a remote live network with a device that you might not have. Since these are pay-as-you-go solutions, it's a great solution to have in your bag of tricks.
3. *Don't completely avoid real devices on real networks.* Finally, there is no way around it; you will need to perform some manual testing with real devices on real networks. But your strategy should be to avoid this option whenever you can because of the expense and the poor diagnostic feedback.

Conclusions

Although you will need to do some manual testing with real devices and real networks, you should seek to avoid this whenever possible. It simply is too expensive, too slow, and lacks the instrumentation necessary for you to isolate application problems so that you can get your product out the door quickly. Instead, consider a solution that combines some manual testing, some remote-manual testing, and a lot of testing using emulated devices.

Emulated devices are very cost effective because they allow you to do a lot of testing quickly and efficiently. But ensure the tool has the diagnostics you will need to isolate problems and the flexibility in network stacks you will need to test different network options. Make sure that your emulated device solution contains a high-level scripting solution to allow you to replay your test cases over and over. Finally, look for an emulated device that allow you to change device profiles quickly.

About Keynote

Keynote is the leading provider of on-demand test and measurement solutions for continuously improving the mobile experience. Content companies will know precisely how their Web sites, content, and applications will perform on actual browsers, networks, and mobile devices long before their customers and business are impacted.

Keynote provides mobile testing solutions under the Mobile Interactive Testing Environment (MITE) family of products. MITE for content testing (MITE/C) can emulate over 1,600 device profiles with 11,000 user agent strings. MITE for on device testing (MITE/D) is a hosted real device testing solution. With the MITE product line, Keynote offers one-stop shopping for all of your mobile testing needs.