macadamian

Test Strategies for Smartphones and Mobile Devices

August 2010

macadamian

Table of Contents

macadamian

Test Strategies for Mobile Device Applications

Why we wrote this paper

Application development for mobile devices is a fast growing phenomenon. According to eWeek.com there were more than 450 million mobile Internet users in 2009, and the number is expected to surpass 1 billion by the end of 2013.

All these users of mobile devices, like iPhone, iPad, Blackberry, Android, the upcoming Windows Phone 7, or even eReaders like the Amazon Kindle, create a never-ending demand for more and more unique and useful mobile applications.

The arena for these new applications is vast, from the healthcare world, where patients can now practice self-care directly on their smartphones[1], to social networking, where 55% of mobile internet users mostly use their devices for social networking and emailing[2].

Users want mobile applications to be simple and fast. Just one nagging bug or usability issue can spoil the entire experience. And with so much competition in this new space, if users don't have an excellent experience with your application, they will switch to a rival product faster than you can say "There's an app for that."

Vendors simply can't afford to go to market with an application that might have a critical bug or usability issue. Yet surprisingly, there is no previously existing comprehensive guide on how to test the particular complexities of mobile device applications.

The strategies presented here are intended to highlight some of the areas where the testing of mobile device applications differs from testing desktop or regular web applications. It is important to plan a test strategy that is mobile-specific, or you may overlook crucial areas of testing like how network connectivity (or lack thereof) affects your application, how screen resolution and orientation changes can spoil a user's whole experience, and whether your application jives with what users of a particular device have come to expect.

We hope you find this guide useful, and encourage you to contact us with questions, comments, and additions.

What you can expect

For simplicity, we divide the testing into four main areas:

1. User Interface Testing
2. External Factors Testing

[1] http://www.diabetesmine.com/2009/03/lifescans-new-diabetes-iphone-app.html
[2] http://www.webcredible.co.uk/about-us/pr/mobile-internet-usage.shtml

3. Stress Testing
4. Security Testing

In some sections, device-specific examples for iPhone, Blackberry, Android, or Windows Mobile smartphones are included. Please explore analogous options for the devices on which you are testing.

Functional testing is excluded from specific mention because that type of testing can be carried out in the same way you would perform it on any other kind of application.

We also include a discussion about testing on emulators versus real devices.

Be aware that testing strategies will depend on the software requirements of the application. Not all testing strategies described in this document apply to all applications. But, as you are creating your test plan, please take the suggestions discussed in this paper into consideration.

User Interface Testing

The first area to explore in your test plan is the user interface. While smartphone applications are relatively new, there are already accepted guidelines for look and feel and overall behavior. It is your job as the tester to confirm that your application follows these principles.

Note that this paper does not cover Usability Testing for mobile devices, which is critical to creating a compelling, usable mobile application. Usability testing and other user-centered design principles will be covered in a future paper.

Comparing your application with native applications

If you are not already a regular user of the device you are testing with, the first thing you should do is familiarize yourself with the device and some of its common native applications such as its Phone, Email, Camera, Contacts, and Calendar programs, etc. These are applications that device users see everyday so it is important that your application has a similar look and feel.

As you are exploring, take note of things like:

1. *Overall color scheme/theme of the device*. For example, Blackberry has several themes[3] that differ even between its own models.
2. *Style and color of icons*. For example, Android has well-defined icon design guidelines[4].
3. *Progress indicators* when pages are loading.
4. *Menus.* How they are invoked and typical items they contain.
5. Overall *responsiveness* of applications on this device.

[3] http://docs.blackberry.com/en/developers/deliverables/6625/Themes_2_3_810712_11.jsp
[4] http://developer.android.com/guide/practices/ui_guidelines/index.html

Screen orientation/resolution

If your application is supported on various devices that have different screen resolutions, make sure you test early with the device that has the smallest screen. This will flush out any layout problems where the design is simply too large to fit on the necessary screen size. Of course, don't forget to do regression testing of any subsequent layout changes to make sure your application still looks good on larger screen sizes as well.

If your device supports screen orientation changes, like the iPhone and some Android, Blackberry and Windows Mobile models do, be sure to include lots of testing where you rotate the device from portrait to landscape display, and vice versa, on all of the pages within your application. This re-orientation completely changes the screen size so layouts may no longer display as intended.

Also, it is easy for developers to leave uncaught exceptions around screen re-orientation events that could cause the application to crash, especially if you do this re-orientation back and forth quickly before the previous orientation change gets a chance to fully complete.

It is also important to test input reactions when the screen orientation is changed. Try using the soft keyboard while changing the orientation repeatedly. On Windows Mobile and some Android devices, a change in orientation is automatically triggered by extending the slide-out keyboard. Attempt this repeatedly and quickly to see if the rapid changes in orientation have a negative effect on the application.

Touchscreens

If your device has a touchscreen, then there are numerous extra things to consider, such as the following:

1. **Multi-touch vs. single touch screens** – If your device and application support multi-touch features, like the pinch-to-zoom effect on iPhone, be sure to include lots of test cases involving touching the screen in more than one place simultaneously, especially while typing on the soft keyboard.

2. **Long touch vs. short touch** – While there is usually no concept of a double-click on touchscreen devices (although there could be if specifically implemented in your application), some, like the Android smartphones, distinguish between long touches and short touches: pressing and holding an item will bring up a context menu in the middle of the screen, while short-clicking the same item will automatically perform the first action in that context menu.

   All Windows Mobile touchscreen enabled devices also use long touch vs. short touch for different reasons: be it to open context menus in certain applications; or to enable a secondary function of a button such as selecting Pause, Wait, and the international dialer symbol when holding down the "*" button in the dialer.

3. **Button size and position** – Ensure that buttons and icons are large enough and far enough from the edges of the screen to be easily clicked by a large fingertip. If the button is not large enough for a

finger to press, such as the Close button on a Windows Mobile device application, verify that a stylus can be used to access and operate such buttons.

4. **Workflow** – Applications should ideally make use of wizards with multiple choice selections like radio buttons and checkboxes to minimize the amount of typing the user needs to do, which can be excessively time-consuming.

5. **Other input methods' interactions** – In most cases your touchscreen device will also have other input methods such as a trackball or hard keyboard. It is important to ensure that virtually simultaneous inputs from all the methods do not interfere with each other. For example, on Android a GUI item may lose its current selected state if you select it by touching the screen, but then switch to using the trackball to navigate.

If you want more touchscreen testing details, please see our 3-part blog series[5] on this topic as well.

Trackballs, trackwheels, and touchpads

If your device doesn't have a touchscreen, it is even more important to verify that screen navigation is as painless as possible for the user. In these cases, the user may rely on a trackball, trackwheel, or touchpad to move from object to object. As discussed in the design section of the Blackberry Development Fundamentals Guide[6], it is necessary for example to check that the trackball is able to move the user from field to field in a logical way.

Even if your device does have a touchscreen, don't forget that some users will still want to use the trackball so it's important to ensure all screen items can be reached with trackball navigation as well. This is an area often overlooked by development teams where inconsistencies can arise.

Soft keyboards

If your device has one, pay special attention to how the user must interact with the soft keyboard. There are often various special cases and corner cases that are overlooked by development, but important to end users.

1. Does the soft keyboard automatically appear if the user's main action is to enter some text?

2. Does the first layer of the soft keyboard include shortcut "@" and ".com" keys if the highlighted field is for entering email addresses? (For example on Android, developers can easily enable this by using the *textEmailAddress* attribute).

[5] http://softwarepmp.blogspot.com/2009/11/qa-strategies-for-medical-touchscreen.html
[6] http://docs.blackberry.com/en/developers/deliverables/1107/fundamentals.pdf

3. Does a long touch on a soft character key bring up several different character choices for that input, such as different accents and symbols on a Windows Mobile device when entering contact information?

4. Can the soft keyboard be dismissed and re-displayed easily?

5. Can the soft and hard keyboards be used interchangeably (if the device has both)?

6. Do soft keyboard characters entered in password fields only show up as ****?

Hard keys

Be sure to include a lot of testing around the use of the device's available hard keys such as Start, Home, Menu, and Back. In order for the user to have a great experience, these should all interact with your application similarly to how they interact with the device's native applications.

For example, Blackberry users expect to make extensive use of their hard Menu key for options, as opposed to clicking buttons or making selections on the screen itself.

Android users will expect a short-click on their hard Menu key to either lead to the application's Settings page or pop up a context specific menu, while a long-click should expose the soft keyboard.

Also note that some HTC Windows Mobile devices have a hard zoom bar below the screen that when slid, should zoom in or out on certain visual aspects of the application.

Shortcuts

Find out if there are any expected shortcuts common for your device and test their use within your application.

For example, you can find a great list of Blackberry keyboard shortcuts on the web[7].

On Windows Mobile devices, the Start Menu will have a list of programs installed that can be launched with a single touch. Newer HTC Windows Mobile devices have the Touch Flo screen appear when you launch the Start Menu, which will display a user editable list of shortcuts to applications. Testing to make sure your application can be correctly added to and, more importantly, launched from this list is critical.

[7] http://forums.crackberry.com/f3/tip-helpful-keyboard-shortcuts-13131/

External Factors Testing

There are numerous other factors that are external to your application itself, but inherent to the mobile device that the application will run on. It is important to also test how these factors may influence your application.

Network connections

Since your application is going to be used on devices in various locations with various network connection speeds, it is important to plan testing coverage for the following scenarios:

1. Only Wi-Fi connection
2. Only 3G connection
3. Only 2G connection
4. With no SIM card in the device
5. In Airplane mode (or all connections disabled)
6. Using the network through a USB connection to a PC
7. And most interestingly, test intermittent network scenarios that a user might encounter in the real world:
    a. Walk out of Wi-Fi range so the connection automatically switches to 3G/2G (for example, in a large building like a hospital or airport, or outdoors)
    b. Ride in an elevator or on a train where the network connection may go up and down
    c. No network connection available at all

SD card interactions

If your application can potentially store or retrieve items on the device's SD card, then it is important to test the application's behavior when there is an SD card present or not. At a minimum, the application should provide user-friendly error messages when a function cannot be performed due to a missing SD card.

Also consider removing the SD card in mid-operation (particularly on devices where it can be easily removed without taking the back off of the device).

Phone calls and other interruptions

If the mobile device you're testing with is also capable of making and receiving phone calls, be sure to test the following scenarios to see how your application reacts before, during, and after the call:

1. Your application is interrupted by an incoming call, originator hangs up the call
2. Your application is interrupted by an incoming call, terminator hangs up the call
3. Your application is interrupted by placing an outgoing call, originator hangs up the call
4. Your application is interrupted by placing an outgoing call, terminator hangs up the call

Also take into consideration such interruptions as:

1. Text messages
2. Voicemail notifications
3. Calendar events
4. Social media notifications (Facebook, Twitter, etc)
5. Alarm clocks
6. Low battery notifications

Any of the above could have an impact on the functionality or overall responsiveness of your application.

Device options

Explore your device's options, and change settings such as the following to see how they affect your application:

1. **Sound profiles** – For example on Blackberry you can change the "Profiles (Ring Tones)" setting to options like Loud, Quiet, Vibrate, Phone Only, or Off. Does your application respect the device's set profile?

2. **Device password/unlock pattern** – You may be able to configure this as a requirement during application installation (for example setting Blackberry's Password option "Prompt on Application Install" to Yes). Does your application still install correctly when prompted for password/unlock pattern?

3. **Font** – How does choosing a different font family, size, or style affect the appearance and usability of your application?

4. **Screen timeout/Auto on, off** – Is your application subject to screen dimming or automatically turning off even when it is actually busy? For example, you wouldn't want your screen to dim or turn off while watching a slideshow of your photos.

5. **Screen orientation** – You may be able to enable/disable automatic orientation switches when the device is rotated, on some Android devices for example. Does your application respect this setting? (However, also keep in mind that the application code can lock a screen to a particular orientation so if you find an issue here it might actually be design intent).

6. **Connections** – Using one of the connections on a device, such as Bluetooth or Microsoft Direct Push (only on Windows Mobile devices), could have adverse effects on your application. How does enabling/disabling Bluetooth or other connection types affect your application's behaviour?

Stress Testing

Although mobile devices' memory and power have improved over the past few years, mobile applications still have much more memory and CPU constraints than desktop applications.

With users expecting applications to hold up even under high traffic conditions with many applications running, stress testing is a must to find exceptions, hangs, and deadlocks that may go unnoticed during functional and user interface testing.

Techniques

The following are some stress testing techniques to try:

1. Load your application with as much data as possible to try to reach its breaking point. For instance, Windows Mobile devices can accomplish this with either ActiveSync (Windows XP) or Windows Mobile Device Center (Windows Vista and newer). This will give the user full access to the Windows file system on the device to add needed testing files where they are required.

2. Perform the same operations over and over again, particularly those that load large amounts of data repeatedly.

3. Perform the repeated operations at varying speeds – very quickly or very slowly.

4. Leave your application running for a long period of time, both interacting with the device and just letting it sit idle, or performing some automatic task that takes a long time (like displaying a 200-photo slideshow).

5. Randomly send screen taps and keystrokes to your application.

6. Have multiple applications running on your device so you can switch between your application and other device applications often, and at different states within your application:
    a. On Android or Blackberry you can press and hold the hard Home or Menu key, respectively, to quickly switch between running applications.
    b. Use all means to launch your application, for example on Windows Mobile use the Touch Flo, the file explorer, and if the application is already launched, maximize and minimize the application.

Tools

There are some tools you can make use of for stress testing on mobile devices. For example, Hopper[8] can send random screen taps and keystrokes to your Windows Mobile application, or try TestQuest[9] for more general automation needs across devices.

Android has DDMS[10] built into its SDK, which among other things can be used to check memory usage.

Non-reproducible issues

Another general tip related to stress testing – do not be afraid to report bugs that you haven't yet found a way to reproduce 100% of the time. If you can at least provide crash logs and a best guess as to how to trigger the problem, that is usually enough to give the developer some ideas on narrowing down the root cause.

Security Testing

General considerations

There are a few things to consider around security at the device level when comparing mobile applications to desktop or web applications:

1. Most mobile devices assume one user; they do not have multiple accounts. (Except in the case of multiple email addresses configured per device, which should be tested especially if your application deals with the device's Contacts or Email functions). But in general there is no concept of user switching, different profiles, or permissions based on user level.

2. It is up to the user whether or not they configure a password/unlock pattern for their device at all.

3. All necessary encryption needs to occur at the application level because the device will not necessarily handle this.

4. Outside communication of any sensitive data should be done over SSL/TLS because it can't be assumed that the user will configure their Wi-Fi to be secure.

For more information on how to perform general security testing on your mobile applications, there is already a good paper[11] so we won't repeat all the details here.

[8] http://blogs.msdn.com/windowsmobile/archive/2007/06/13/how-to-run-hopper-on-your-application-step-by-step-procedures.aspx
[9] http://www.testquest.com/index.cfm
[10] http://developer.android.com/guide/developing/tools/ddms.html
[11] http://www.foundstone.com/us/resources/whitepapers/wp_mobile_application_pen_testing.pdf

Web applications

While running most web security testing tools on a mobile device is virtually impossible, you can run your mobile website on a PC itself, by following the instructions here[12]. Then you will be able to test the application with common web security testing tools, such as Firefox Web Developer add-on, SQL Inject Me, XXS Me, WebScarab, or Brutus.

A web proxy can also be used to intercept all mobile device traffic to monitor data and test for security issues.

Sensitive information

The threat of passwords or usernames being left unencrypted on the device is a real one that can be taken advantage of by other users.

To test for this, footprint and fingerprint analysis can be used to make hash files of the device file structure, with a tool like md5deep (as described on page 5 of the following paper[13]). Then use a difference comparison application such as ExamDiff to compare the before and after hash files to see which files have changed. By doing so, you can then search the files your application changed for unencrypted information that could be used to compromise the device.

Application and device permissions

During installation of your application on a mobile device you are likely to encounter options for application execution permissions. For example, on Windows Mobile devices there are three levels: Privileged, Normal, and Blocked, which are specified by the application's certificate, and define how much access the application will have to the device itself. On Blackberry, you may be asked to allow, prompt for, or deny your application access to various connection types, interactions, or user data on the device.

Research the permissions configuration on your device and design your testing to ensure that your application can get access to the device areas it needs in order to function properly, and it provides useful error messages directing the user to set the permissions properly if it cannot.

The device itself could also have different security configurations, such as security off, various levels of prompting, required third-party signing, or locked. It is important to verify that your application can be successfully installed at all expected device security level settings. For example, on Windows Mobile a high security level may not allow an unsigned CAB file to be installed, even if it is wrapped in an MSI file. This kind of behavior must be discovered and mitigated.

Even after installation, it is possible to change the security settings of your device or application permissions. You should include some test cases to cover these scenarios as well.

[12] http://www.aswinanand.com/2009/01/open-mobile-websites-on-pc/
[13] http://www.foundstone.com/us/resources/whitepapers/wp_mobile_application_pen_testing.pdf

Testing with Emulators

Emulators can be very useful to cover a breadth of devices (with different screen resolutions for example) to which you may not have physical access.

However, it is important to keep the following caveats in mind:

1. Not all activities can be realistically emulated, like switching network connections, or taking a picture or video.
2. Some activities just don't work at all on emulators, like streaming video on a Blackberry emulator.
3. Due to lower device power and memory, your application could exhibit slower performance overall when run on an actual device (versus in an emulator on your powerful desktop computer).
4. If the emulator and actual device have different dpi resolutions, your screens may not display as you expect.

Because of these and other more subtle differences, be careful not to assume that just because your application works perfectly on an emulator, it will have no issues on a real device. If possible you can initially test your application simultaneously on a real device and its analogous emulator. Take note of any discrepancies seen between the two and base your future real device testing plan on the differences observed.

In general, it is a good idea to use some combination of real device and emulator testing, as recommended in the table below:

Type of Testing	Manual Testing		Automated Testing
	Using Devices	Using Emulators	
Unit Testing	No	Yes	No
Integration Testing	No	Yes	No
System Testing	Yes	No	No
Regression Testing	Yes	No	Yes
Compatibility Testing	Yes	No	Yes
GUI Testing	Yes	No	No
Performance Testing	Yes	No	Yes
Security Testing	Yes	No	Yes
Synchronization Testing	Yes	No	No

There is an excellent guide for obtaining and using emulators for testing on various mobile device types[14]. It covers iPhone, Blackberry, Palm, Windows Mobile, Android, Symbian, and Nokia emulators.

[14] http://mobiforge.com/testing/story/a-guide-mobile-emulators

As an alternative to both emulators and buying lots of physical devices, you could also consider using a service like DeviceAnywhere.com that gives you online access to numerous real devices on various networks.

Conclusion

When planning your testing effort for a mobile device application, in addition to the usual functional testing, it is also important to consider the following areas and how they differ from desktop or regular web applications:

1. **User Interface Testing** – mobile devices have unique user interfaces like smaller screens that can be re-oriented, touchscreens and soft keyboards, and navigation methods like hard keys and trackballs.

2. **External Factors Testing** – mobile device applications must also contend with interactions and interruptions from other device features like various network connection types, SD cards, phone calls, and assorted device settings.

3. **Stress Testing** – mobile device applications have much less overall device memory and power available so must handle themselves very efficiently.

4. **Security Testing** – mobile device security will become more and more important as the user base grows, so it is essential to test the security of your mobile web applications, sensitive data storage, and how your application behaves under various device permission schemes.

5. **Emulator Use** – Emulators can be a great asset when it comes to achieving testing coverage on multiple devices, but your test plan must also respect the fact that sometimes there is just no substitute for the real thing.

About the Authors

Tanya Dumaresq is a Quality Assurance Manager at Macadamian, and an expert in test automation, test case design, and testing strategy. She is a regular speaker at the STAREast and STARWest conferences and her articles about software testing have been published in Better Software and SD Times. Tanya can be reached at tanya@macadamian.com.

Matt Villeneuve is a Quality Assurance Specialist at Macadamian, and an expert in rapid timeframe testing and testing on projects with short delivery cycles. His background includes testing both hardware and software products, and he takes a holistic view of software testing, working closely with the development team and user experience team to insure that the complete set of the customer's requirements are being met. Matt can be reached at matthew.villeneuve@macadamian.com.

About Macadamian

We're a User Experience Design and Software Development consultancy that works with ISVs and technology companies to design and develop compelling software products. We have a special focus on mobile, helping product companies extend their products to mobile devices, and create new products for mobile platforms including the iPhone, iPad, Android devices, Blackberry, and Windows Phone 7.

Let us know what you think of this white paper, and feel free to get in touch if you have any questions. The best way to reach us is at software@macadamian.com. You can also visit our website at www.macadamian.com

Glossary

Acronym	Full term
CAB	Microsoft Cabinet file
DDMS	Dalvik Debug Monitor Server
HTC	High Tech Computer Corporation
MSI	Microsoft Installer file
PC	Personal Computer
SD	Secure Digital
SDK	Software Development Kit
SIM	Subscriber Identity Module
SSL	Secure Socket Layer
TLS	Transport Layer Security
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity