Test Mobile Applications with I SLICED UP FUN!

BY JONATHAN KOHL

A MNEMONIC FOR MOBILE APP TESTING

Test idea generation is the cornerstone of effective software testing. It's hard to think of different approaches when testing software, so an easy way to get started is to use a mnemonic to help guide your thinking. I call it: I SLICED UP FUN!*†*

This is a mnemonic I adapted from James Bach's SFDPOT (San Francisco Depot)[1], a testing mnemonic that explores product elements. Each letter in the mnemonic represents a different approach to testing mobile apps. I have found each of these areas to be effective at revealing problems with applications I am testing. Try using the app on your test device, focusing on each of these areas, one at a time. Spend time in each category to think of and execute different test ideas.

Here is I SLICED UP FUN! expanded:

I: INPUTS INTO THE DEVICE

These represent the ways you can interact and control the device. Including:

- Built-in keyboard/keypad
- Touch screen gestures and typing
- Synching with other devices
- Peripherals that you can plug in to the device
- Hold the device differently when inputting– be creative!

Test the application and try out all the inputs that you can think of. If it is a touch device, make sure you try different combinations with your fingers. Keep a thumb on the edge of the screen and try to type, or tap your fingers on the device while using it. If it rotates to portrait and landscape modes, make sure you try in different modes. See if the application behaves strangely when you rotate it and try to manipulate with touch gestures or typing on a keyboard.

Even the way devices are held can impact their connectivity and how well they respond to application inputs. Can you hold the device in such a way and interact with it so that it causes the app you are testing to crash or freeze up? What happens if you launch the application in each supported orientation of the device? (Sideways, upside down, straight up, etc.)

S: STORE

Mobile apps tend to get distributed online through special stores. The most well known are Apple's App Store and the Android Market. For test ideas, try to find out information about the requirements of application requirements for store submissions. Sometimes finding out what store(s) the app will be submitted to and looking at any public documentation available that outlines guidelines can be a treasure trove of testing ideas.

See if you can find anything that might have been missed by looking for:

- Submission specifications
- Development guide

- User guide for error handling, location services, permissions for user privacy items, accessibility, etc.

L: Location

Where you are located can have an impact on what you are testing, particularly if the application requires an internet connection. Applications may be affected by the following:

- Geo-location errors
- Movement, stopping suddenly
- Connection issues due to interference
- Moving from one data network to another (eg. wifi to wifi, wifi to wireless broadband, wireless broadband to wifi, others)

Getting up and carrying your device while testing an app is an effective way to find bugs. Just get up, walk out of range of the wifi device you are connected to, and see what happens to the application when you move from one network to another wifi or other network. Can it handle your devices native messages when you change or lose connections? Does it crash or freeze if it loses or changes connections?

Also, your location is often different from where the application is developed. You may speak a different language than the programmers. Is the application compatible with the language and culture where you are located? Are words correctly translated? Are meanings correct and clear? Does the application do something that could be interpreted as funny or offensive? If so, the development team needs to know those issues.

I: Interactions/Interruptions[2]

Mobile applications have far more hardware restrictions when they operate than other computers. They have less memory and less processing power than a PC, for example. As a result, they are susceptible to problems when other applications or operating system functions are running. It's important to test how the app behaves when other applications running at the same time. Furthermore, any changes made to the device's preferences can have an effect on your app. See how interaction with other programs, particularly built-in, native applications and the application you are testing interact. On mobile devices, they can cause your app to fail. You have to be very conscious of your memory footprint and how you shut down and clean up.

Are there problems when:

- Running multiple applications, utilizing multitasking
- Using other applications, then using the application you are testing(email, calendar, texting, note taking, others)
- Notifications appear (new emails, phone calls, text messages, other notifications)
- Error messages occur (losing connections, notifications, operating system and other errors)

Be creative with your device settings. Find out all the ways you can change the settings on your phone, and determine which might have an effect on the application. Spending a couple of hours using the application while changing different settings is worthwhile. Also spend time exploring ways you can force error messages.

Another interesting interaction is with any time-based notification, such as calendars, alarms, and built-in clocks. Applications can behave strangely when they become confused about time, or if they are interrupted by a time-related alert. Even using applications in a certain order, or doing things on certain days (time changes for example) can cause interruptions or interactions that the application may have problems dealing with.

C: Communication

Smartphones started out primarily as communication devices, but the app we are testing may not handle communication interaction smoothly. Here are some communication examples:

- Phone
- Texting

- Emails
- Instant messaging
- Voicemail messages
- Video
- Others

It is interesting to test how an app interacts with communication: texting, emails, telephone, voicemail. There are a lot of combinations to test here: taking calls/texts, rejecting calls/texts, interacting with voicemail. Each of these communication devices take precedence over the app you are testing, so how does it handle these interruptions? There are a lot of scenarios you can try. In some cases, you might be able to use a communication device while using the app you are currently testing.

E: Ergonomics

If you test mobile apps for any length of time, you will show signs of physical stress. You need to manage this as a tester, by taking breaks and managing your time and physical interaction. However, also think of the end users. They run into the same problems that can make you tired:

- Small screens can be hard on the eyes
- A small device means there is no ergonomic help from a desk, or chair – you often hunch over to interact with it
- It's not uncommon to get a sore back, fingers, eyes when using a device for any length of time
- Use physical strain as points of investigation for usability problems

Use physical cues from your own testing work and analyze whether the application makes interaction worse. Are fonts and colors and sizes too small? Are there too many steps to click through to get through complicated workflows? Does the end user have to type a lot on the small device? Any shortcuts to decrease physical strain, or improvements to display are huge factors with how successful an app is. Highlight and report problem areas that make you struggle – users will have the same kinds of problems if they use the app for any length of time.

D: Data

The "D" in SFDPOT stands for data, in other words, anything the application processes. Anything you can input into the application is processed in some way. Furthermore, any data that is coming from a server or third party can have an effect on the application. Check for:

- Types of input – special characters, different languages, etc
- Media – see if the app depends on an outside source to play music, videos or anything else.
- Size of files – if the application uses outside files, try using different file types
- Frequency of updates – some applications require data from a server that is updated periodically to change settings, or to provide new content. Find out the schedule of these updates and see how the application reacts when an update occurs.

U: Usability

These can be some of the most important bugs to find and log, because they are the issues that often frustrate end users the most. When you are testing the application, note and log any issues that make you uncomfortable, frustrated, angry, or upset. To find usability issues, look for any actions that are awkward, confusing, or slow. Listen to your emotions – that's an important source of usability issues. Instructions can be incomplete or misleading, items can be labeled incorrectly, and there may be workflows that make doing what you want to do difficult. If there are aspects of the application that you don't like and frustrate you, log them as usability issues. Sometimes installation, setup or upgrading an app can have complicated steps. Are there any strange configuration interactions with your phone's settings or hardware? Some device updates require switching power off, removing batteries, and other things that may be difficult for non-technical users to do. Highlight anything cumbersome, frustrating or confusing for the development team.

P: Platform

Borrowing the "P" from Bach's SFDPOT mnemonic, looking at the device that the application needs to run on is important. You may have a device that the programmers do not have access to, and their app may not work at all, or may work poorly. Your unique device is something that they are depending on you to test the application, so careful observation and documentation of problems, large or small is important. Also be aware of changes to your device. You have little control over operating system updates. The provider encourages you to update these, for good reasons. However, new versions can cause an app to fail that worked in a prior version. Be aware that updates (even small ones) can cause an application to stop working. Be sure to find out all the technical information about what device you are using, and what operating system version you have installed when logging a bug.

F: Function

Functional testing (the "F" in SFDPOT) is the most common form of testing. It usually involves comparing the specification or requirements for an app, and verifying that the application does what it says it does. This can often be very simplistic: looking for items and ensuring they are there and function the way they are supposed to. To be more creative, generate test ideas by asking these questions:

- Can you identify everything that the application does?
- Have you worked through all the aspects of the app? Clicked every button? Filled in every form? Have looked into default settings and options that are available to you?.
- Try a tour of the product to identify everything it does[3]. Try to find commonly used features, and the most obscure. Be thorough.
- Once you have a good sense of what the product is supposed to do, does it match what the app developers say it does? If it doesn't, there is a source of problems. If your impression differs from theirs, they need to know that.

U: User Scenarios[4]

This is an incredibly powerful method of finding problems. In fact, if I am limited to one type of testing approach in a short period of time, this is often the one I will choose. (Check out Cem Kaner's paper on scenario testing in the references to help get started.) This is a fun way to use the application: imagine two or three real users that you know, and think what they would do when using the application. Try to think of creative users: I always imagine that one annoying family member who struggles with technology and always wants me to fix their computer or mobile phone. If I pretend to be that person, what kinds of problems would they find? What would they struggle with? I also imagine a more technical user, and someone in between. I try to think of real people and how they would react to using the software. To create scenarios, I identify the following

- How is this application supposed to be used?
- What problems does it solve for users?
- What are the goals of end users that this application helps them solve?

I then create scenarios based on that information. For example, if I think of a busy office worker, I set up a scenario with my phone so that it has a lot of emails, texts and meeting reminders occurring while I use the app. You can get very creative here, with a lot of other application interaction, as well as trying to quickly solve a problem or enjoy an experience with a device under different conditions.

N: Network

Mobile apps that require network connectivity such as an internet connection are quite susceptible to problems. Since they aren't stationary objects like a PC, they have a huge dependence on the availability, performance and reliability of networks. The path from a server to your device can be confusing, complex and circuitous. What this means is that the application may expect connectivity that isn't available consistently where you are testing. In these cases, the application may fail or crash because of timing issues (it takes too long to get responses) poor

connections (lack of signal strength) or moving from one network to another.

- Wifi
- Wireless broadband
- Dead spots
- Moving from one data network to another (eg. wifi to wifi, wifi to wireless broadband, wireless broadband to wifi, others)

The network controlled by a service provider and how well it performs is not something we can control. We can find out whether apps work well with the network we are using or not, and let the developers know about the problems you see.

GETTING STARTED

I usually start with "FUN" – I look at the functions of the app, starting with the installation and moving from there. I follow any instructions (or lack of) it comes with, and spend time touring the application. I then look at all the settings options of my mobile test device and determine what might interfere with the application. It can take over an hour to delve deeply into the built-in capabilities of the device itself. I then try to think of three different users, and imagine how they would use the application. I map out basic scenarios, and try to move my way through the application the way real users would. As I move through the app, I am careful to note anything that bothers me, and feels unresponsive or awkward. These are usability issues that are important feedback. Then, I investigate my network options, and I try the app out using different connection options. From there, I move on to the rest of the mnemonic: interactions, interruptions, and others. You might be surprised how many bugs you find if you work at trying out as many ideas in each category as you can. Don't be limited by this though, come up with your own ideas, use this mnemonic as a springboard for your testing thinking.

Note: The ideas represented by I SLICED UP FUN! are there to help get you started, but they are by no means exhaustive. ***I recommend using other models to help generate test ideas, particularly in the areas of performance and security***, which I don't touch on here. If you think of other ideas, be sure to try them out, and use this concept in conjunction with other areas you want to focus on in your testing project.

This article was first published by Kohl Concepts Inc., October, 2010.

† Thanks to Jared Quinert for helping create the I SLICED UP FUN! mnemonic.

REFERENCES

[1] Bach, James. (2006) Heuristic Test Strategy Model http://www.satisfice.com/tools/satisfice-tsm-4p.pdf p. 4

[2] Testlabs Blog. (2010) Top 10 Tips For Testing iPhone Applications http://blog.testlabs.com/search/label/iPhone accessed September, 2010.

[3] Kelly, Michael (2005) Touring Heuristic http://www.testingreflections.com/node/view/2823

[4]Kaner, Cem. (2003) An Introduction to Scenario Testing http://www.kaner.com/pdfs/ScenarioIntroVer4.pdf