Addressing mobile applications risk:
A software quality focus

kpmg.com
Addressing mobile applications risk: A software quality focus

Mobile device users have grown accustomed to accessing mobile applications (apps) for just about any use or interest. In recent years, their expectations and enthusiasm have translated quickly into big numbers: consumers worldwide downloaded more than 29 billion apps in 2011, up from 9 billion in 2009.¹ What’s more, International Data Corporation (IDC) predicts that mobile app downloads will exceed 182 billion globally by 2015.²

Like individual consumers, large companies have also embraced mobile apps, which help them enable employees worldwide to access enterprise applications and business platforms from their mobile devices. By 2014, 90 percent of organizations are expected to support corporate applications on personal devices.³

Consequently, developers now face fierce competition in a crowded marketplace for consumer-focused mobile applications. Mobile app development is growing rapidly, but it differs in many ways from the development of traditional applications in large enterprises. These differences create new business risks for developers.

For example, companies expect to deploy mobile apps much more quickly than other applications, either by using agile development methodologies in-house or by outsourcing development to specialized providers. In addition, mobile apps often have limited life spans, which means that complete rewrites or redesigns are commonplace across versions and iterations, often under deadline pressure.

Marketplace stakes are too high to overlook mobile app quality in the interest of time to market. Users’ feedback is easily accessible in the app marketplace(s) and influences customer purchasing behavior. Quality issues will be broadcast to other potential customers and poor feedback/low ratings could negatively influence future customer purchases and potentially damage a company’s brand.

This paper provides a perspective on the specific software quality attributes to consider when an organization formulates a risk-based approach to mobile application software testing.

Organizations may use a variety of software development methodologies to build mobile applications; this paper presents recommendations aligned with the activities in KPMG’s System Testing Services methodology and the traditional waterfall development methodology. It does not cover certification testing for application storefronts/marketplaces.

¹ ABI Research, Mobile Applications Market Data, January 2012, http://www.abiresearch.com/research/1004105?ll&viewtable=1009205-MD-APPS-108.xlsm-Table2.csv
Software quality attributes and risk-based testing

A risk-based approach to software testing prioritizes the testing of software systems based on the risk of failure. As with testing Web applications or server applications, testing mobile apps should rely on a risk-based approach. If an aspect of the system has many risks, more thorough testing is needed; the opposite is also true: “less risk, less test.” These system aspects can be categorized in the ISO 9126 software quality model and presented in the figure below.

The inherent risk(s) associated with a mobile app will vary for each mobile application based on functionality, scope, and intended customer/use. The application stakeholders, development team(s), and quality assurance teams/organizations are responsible for identifying and prioritizing appropriate risk areas. A software testing organization should focus on the following software quality attributes when they formulate a risk-based software testing approach:

- **Interoperability** – Numerous applications and services may run simultaneously on a mobile device. Testing must address how the mobile application should respond when other services are active (e.g., an app requires the sound output currently in use by a music player app or during an incoming phone call).

- **Recoverability** – Mobile devices rely heavily on their batteries; thus, mobile app testing should consider related power limitations. Also, modern mobile operating systems tend to do a poor job of managing concurrent services and applications. Mobile application testing should focus on how the app handles unexpected events.

- **Efficiency** – Mobile apps are expected to be efficient in their consumption of limited battery power. Today’s mobile devices boast a myriad of power-hungry components including GPS, multiple network radios, and video cameras. Some providers may not offer sufficient battery capacity in their devices to consistently power these components, and they may even cause an application crash. Mobile app testing should focus on efficiency of power consumption.

- **Security** – Mobile devices house considerable personal information such as contacts, account passwords, and credit card details. Too often, however, mobile apps have been tested to rely on unsecure connections, store data unencrypted, or allow direct access to data through an unsecure interface. Testing should address these potential security issues through the use of application and network security testing tools.

© 2014 KPMG LLP, a Delaware limited liability partnership and the U.S. member firm of the KPMG network of independent member firms affiliated with KPMG International Cooperative (“KPMG International”), a Swiss entity. All rights reserved. Printed in the U.S.A. The KPMG name, logo and “cutting through complexity” are registered trademarks or trademarks of KPMG International. NDPPS 275608
• Fault tolerance – Mobile devices are, as the name suggests, mobile. With mobility comes new problems, such as suspended connections, varying network strengths, or (temporarily) no GPS signal. Mobile app testing should consider these challenges and understand the accepted tolerance thresholds.

• Usability – User experience of the application differs by device. For example, scrolling to a “Submit” button at the bottom of a page is very different on a touchscreen-based smart phone Apple iPhone compared to the same action using a nontouch device. Mobile app testing must account for these differences.

The following sections will further describe the software quality attributes from a mobile app perspective and discuss testing techniques and considerations to mitigate the potential risks.

Interoperability

Interoperability testing evaluates an application’s ability to interwork and cooperate in a given environment. As mobile hardware, platforms, and mobile apps proliferate, mobile apps must coexist successfully on a variety of new devices. A mobile app must not negatively affect a device’s performance or interfere with other applications or processes. The reverse holds true as well—the new mobile app should function correctly while other applications/processes are running simultaneously. Interoperability is an aspect of application functionality. Functional attributes of an application refer to those that contribute to its specific functions and serve specific requirements.

Mobile app interoperability is a larger risk for publicly deployed applications vs. those deployed in an enterprise environment, where policy restricts mobile device configurations. However, interoperability issues can still surface when companies allow their employees to use their personally owned mobile devices to access company resources such as e-mail, enterprise applications, and restricted/proprietary content/information (a practice that has been termed “bring your own device” (BYOD)).

Traditionally, a test organization would formulate a test approach to address interoperability for a specific hardware and software configuration that matched the production environment. But the variability of hardware and software in the marketplace means that effectively addressing interoperability risks by testing a single configuration is virtually impossible. Time and cost considerations limit the feasibility of testing all configurations, so the choice requires considerable care and consideration.

When testing traditional applications in a large enterprise, the application requirements dictate the hardware/software platform, and the test environments are built to match these requirements. The environment’s scale and scope are adjusted based on test level. The challenges in a large enterprise are typically procurement, installation, and support of test environments.

Different challenges emerge when testing mobile apps. Like Web apps, mobile apps are designed for a specific mobile platform (i.e., Android, iOS, or Blackberry), and multiple unique devices run this platform. Each device has varying characteristics (e.g., screen size, processor speed, WWAN radio, RAM, input style [touch, QWERTY keypad, etc.], storage, etc.). Testing every configuration is impossible, so developers need to make sure they test their apps on devices that represent their target user base.

These challenges should be addressed during the requirements/design phase(s) of the development life cycle by identifying the supported platforms (iPhone/iOS, iPad/iOS, Blackberry, Android, etc.) and the supported OS versions (iOS 3.13+, Android 2.3+, etc.) to be tested. These are basic requirements for a mobile app and will narrow the device pool to a subset that can be further refined by utilizing a combination of market share data and mobile device criteria.

Recoverability

The physical, hardware, and software environment for a mobile device is unpredictable and far different than the climate-controlled data centers where traditional enterprise applications reside on large servers. The irregularity and variability of mobile environments introduce conditions that can impact the functionality of a mobile device and potentially cause its mobile apps to fail. These risks underlie the need to test for application recoverability, which is defined as the ability to bring the system back to normal function after failure.

Common failure scenarios include an unexpected power-down due to the needs of power-hungry background processes/applications or the unseating of the battery when the device is dropped. In either scenario, the tester would validate whether the application restarts properly and if in-app data/settings are retained. Executing this scenario appears simple, but in reality it is labor and time intensive because batteries must be drained or physically manipulated to reproduce the scenario on each device.

Physically manipulating mobile devices is a recurring requirement for mobile app testing and is somewhat comparable to physical keystroke/mouse manipulation requirements for traditional desktop application testing. In the desktop setting, the solution is to leverage an automation
toolset to capture and playback keystrokes for each test case. There are automation testing tools available for mobile app testing that provide this functionality. These tools can automate testing on physical mobile devices, mobile emulators, and virtualized devices. Many of these tools can also replicate specific hardware/software conditions (e.g., low battery, battery failure, network connectivity, etc.) to test for quality attributes like recoverability.

**Mobile emulators** – To help reduce the cost and complexity of mobile device testing, mobile emulators allow the tester to simulate a mobile environment on a desktop PC. Emulators can be leveraged for unit testing by development teams or for quickly testing key functionality in new application builds. They allow for expanded test coverage and ease of use in the absence of a real device. There are three types:

- **Device emulators** – Typically developed by device manufactures to simulate the actual mobile device, the device emulator provides a virtual representation of the device’s screen and buttons. Testers should follow the device manufacturer’s guidelines for installing and utilizing an emulator, and pay specific attention to the limitations of the software.

- **Browser emulators** – By simulating a mobile browser that would run on a mobile device, emulators are useful for testing mobile Web apps but lack any device-specific characteristics.

- **Operating system emulators** – Mobile device operating system developers including Google, Apple, and Microsoft provide desktop software applications that simulate the device environment and provide access to applications running on the device.

**Virtualized mobile devices** – Several third-party firms provide remote controllable access to mobile devices via the cloud. These devices are also accessible anywhere and are well suited for geographically distributed testing teams. Additionally, many leading test automation tools can “plug in” to this interface for record/playback of automated test cases. The test organization can leverage this library of devices to extend its test coverage when testing is not feasible with lab-based physical hardware or emulators.

**Test automation tools** – These tools function by recording user interaction and playing it back to measure actual vs. expected results. They are best used for regression or smoke testing of an application to quickly assess basic functionality in new builds or versions of software. Test automation tools are known for their ability to test desktop applications, and they function well with desktop-based mobile emulators by “recording” user interaction (keystrokes and mouse movements) and playing it back as a test case. Other types of tools and technology are available that allow for record/playback on actual mobile devices via special hardware interfaces.
Efficiency

The nature of a mobile device requires that it function untethered from an electric outlet, so well-designed mobile applications do not use excessive power. Testing a mobile application’s battery resource consumption is called power management testing.

Proper power management testing requires measuring the power consumed by a mobile app on a device. Power/battery consumption can vary across mobile devices. To account for this variability, it is critical to perform numerous power management test cycles to generate a large pool of results that can be statistically analyzed. The tester can leverage automation tools to execute multiple test cycles and should use physical devices to gather actual battery/power consumption data.

Prior to measuring the power management characteristics of the mobile app, the mobile device’s power management data should be monitored in suspended, idle, and active usage states:

- **The suspended state** is the most typical for a mobile device. In this state the device will consume the least amount of power: the processor is idle and only the communication subsystem is active, so the device can receive incoming voice calls, text messages, and other data.
- **In the idle state,** no applications are running but the device is fully “awake,” so that the active display, which can consume up to 50 percent of the battery capacity, is operational.
- **The active usage state** occurs when the device is actively engaged in a voice call, data transmission, or running a mobile app.

There are two approaches to evaluating mobile application power management: (1) component level and, (2) device level.

1. The preferred approach is device level power management evaluation because it is simpler and more practical for testing in software development organizations. The tester should begin by compiling baseline power management readings for the device in suspended, idle, and active usage states. As noted above, the tester should take multiple readings to help ensure statistical significance and overcome variability in device power consumption. Once the baselines are set, the tester should measure power management during the execution of typical use scenarios. Again, the recommendation is to execute multiple test cycles with automated testing tools to gain statistical significance from the test data. The mobile app’s power management data should be evaluated against the mobile device baseline data to determine if the app’s power management characteristics are acceptable.

2. Component level evaluation relies on gathering specific power consumption data during the execution of specific mobile app scenarios. The tester uses specialized test equipment to measure power consumption from various components, such as the radio, audio speaker, CPU, and display. This approach is more detailed than the device level approach and requires a high degree of technical skill and knowledge along with the appropriate equipment. It is best suited for mobile device manufacturers and OEMs.

Bandwidth consumption is another key area for mobile app efficiency. Since most U.S. mobile device users have limited data plans, an application that uses excessive bandwidth can cost a company significant amounts of money in overage fees. Specific techniques for testing bandwidth consumption can be employed during the design and development stage of the life cycle. This is a specialized area that requires network analysis tools and a knowledge of how the application makes use of data.

Security

Security breaches make headlines for large companies, and analyzing security risks after an incident occurs is too late. Software security is a wide-ranging topic for mobile applications but generally refers to the mobile app’s ability to prevent unauthorized access to programs or data. Security risks for mobile apps go beyond those of a standard desktop application; unique concerns include the loss of a device, exposing access/data to mobile apps; employees using their devices in public/unsecured Wi-Fi hotspots; and mobile malware.

There are two main categories of mobile app risks:

- Malicious functionality, which is unwanted and dangerous behaviors that are placed in a Trojan app that the user is tricked into installing
- Vulnerabilities, which are errors in design or implementation that expose the mobile device data to interception and retrieval by attackers.

However, security risk varies depending on the type of mobile application and the sensitivity of its information. For example, a mobile app that displays local movie times would have low security risk compared with an enterprise application that displays private corporate financial data or trade secrets.

Developing a mobile app security testing capability within a traditional software testing team is a challenging task unless the team is already proficient in security testing for Web applications. Experience shows that traditional
software testers might not have the proper mindset, says James Whittaker, noted software security expert, in his book, How to Break Software Security:

“Software can be correct without being secure. Indeed, software can meet every requirement and perform every specified action flawlessly yet still be exploited by a malicious user. This is because security bugs are different from traditional bugs. In order to locate security bugs, testers have to think differently too.”

When the stakes are high and private data is at risk, security testing is simply not for the novice. The testing team should consider engaging an outside security testing firm to handle this testing and work to develop this capability in-house over time.

Fault tolerance

Fault tolerance refers to the application’s ability to function properly when some of its components fail. Fault tolerance is typically a high-risk software attribute for high availability applications, such as credit card processing systems or life-critical systems in a healthcare/medical setting. Mobile applications require a high degree of fault tolerance as well, but for different reasons. Mobile devices routinely experience “faults” in connectivity. For example, the mobile device may lose connectivity to the mobile network or to GPS in certain geographic areas due to limited coverage. Mobile applications must be designed to properly function in the absence of this connectivity because it would be unacceptable if the mobile device “switched off” or crashed every time it lost a wireless signal. If the mobile application under test does not require network connectivity or other wireless technology (e.g., GPS, NFC, Bluetooth, etc.), the fault-tolerance risk may be lower.

The tester can leverage testing tools to simulate faults and then evaluate the impact on the mobile app. Virtualized device providers feature tools that allow testers to simulate transient network conditions during test execution. The tester should identify specific test scenarios that utilize wireless connectivity and execute them under varying network conditions to evaluate fault tolerance.

Usability

Mobile application usability testing evaluates usability, ease of use, and “learnability.” Usability risks are high for mobile apps due to their new and nonstandard or unfamiliar interfaces. Usability issues also arise due to the large variety of mobile device configurations. An application designed to work properly on an iPhone, for example, may not be easy to use on a touch-screen Blackberry or Android device.

Usability testing must be done with a physical device and executed manually by a user; to avoid bias, this person should not be a developer, tester, or anyone who may have contributed to the mobile app’s development. The traditional usability testing approach leverages large-scale surveys distributed to application users, who are asked for feedback about the application’s ease of use. This approach is relatively low in cost, automatically deployed, and capable of gathering large amounts of feedback. However, there are downsides: most users have short attention spans and do not complete long surveys. Additionally, the survey mechanism is most proficient at capturing and evaluating quantitative data—users rarely fill out the free text fields to explain why they answered a certain way and when they do, the answers are usually difficult to evaluate. Finally and most important, the survey approach can only be used by an existing mobile app that has been deployed to production or the marketplace. It does not address the usability risk for newly developed applications.

To overcome these limitations and target actual users for usability testing, software-testing organization can engage a crowd-sourced, usability-testing provider. Crowd-sourcing is defined as “the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call.”

The crowd-sourced team is unbiased: they are not developers or employees within the software development organization and are focused on providing high-quality usability testing feedback because they are being hired to do so. The provider can ramp up/down usability testers based on need and cost considerations. Additionally, testing across multiple mobile device configurations from a crowd-sourced base is more cost effective than procuring the hardware in-house or soliciting surveys to users in production. Finally, newly developed applications can be distributed to the pool of testers without deploying to production. Crowd sourcing’s strength, a readily available lower cost workforce, is often cited as a potential weakness because cheap nonprofessional labor might deliver a less credible product. The usability testing plan must be simple and straightforward; anything beyond this approach is too risky for crowd-sourcing. The software testing organization should formulate a robust crowd sourcing management strategy to coordinate the large workforce and consider using a service provider that specializes in crowd sourced testing to avoid overburdening their organization with an unfamiliar task.

Conclusion

Consumers have sent a clear signal to mobile app developers—they love mobile apps and are downloading them at an accelerating rate. There are low barriers to entry for mobile app developers to create and distribute their applications, but the new app marketplaces have become crowded with apps of varying quality. Testing cannot be an afterthought—risks are too high. Consumer feedback occurs in real-time and is a key contributor to purchasing behavior. A risk-based approach is a critical factor in helping ensure the success of mobile apps in the marketplace and it is important to focus on different software quality characteristics for mobile apps. New testing techniques and tools are needed to help companies quickly develop and redesign secure, stable, functional mobile apps. Such techniques and tools must above all help the developer manage and mitigate the business and operational risks specific to mobile apps.
