IBM mobile quality point of view

Executive summary

The majority of mobile apps are multitier architecture. The code that runs on the mobile device itself is the front-end client for data and services supplied by more traditional middle-tier and data center back-office systems. Effective and comprehensive testing of mobile apps requires that all aspects of the application be addressed, not only the code on the mobile device.

The implication of comprehensive, multitier testing is that multiple tools and test capabilities must be used and coordinated to create a quality result. No one testing technique is adequate for mobile apps. This paper describes the IBM point-of-view about which execution engines are necessary and appropriate for a complete mobile testing solution.

Currently, several techniques for testing and validating mobile apps are being used in the market place. The IBM point-of-view is that an effective development project employs all applicable techniques for their mobile app quality assessment because each technique has its strengths and weaknesses. The various techniques available are not mutually exclusive. The most effective testing strategy balances the use of all forms of mobile test execution and rolls up the results from each into an extensive, overall "quality metric" for mobile apps.

Automated versus manual testing

Some mobile app testing is automated and unattended, while other testing must be interactive and manual. The best quality assessment of your mobile app employs a balanced combination of both automated and interactive testing.

Automated mobile app testing is critical for accelerating delivery of your application and maintaining the velocity of your mobile app development lifecycle. A wide variety of automated testing techniques can be applied to mobile apps, and each technique has certain strengths. Therefore, the following mix of automated mobile app testing techniques is important:

- Random generated mobile tests
- Keyword-based mobile app test scripts
- Programmatic user interface testing applications (UIAutomator/UIAutomation)
- Behavior-driven development (BDD) testing
- Image recognition-based automation
- Instrumented application object and event-based automation

However, automated testing of your mobile application is insufficient for ensuring the best quality app. Certain aspects of the quality of your app cannot be determined with automated testing techniques alone. The "look and feel" of the app, its usability, its logical flow and other more subjective aspects are a better fit for interactive manual testing and assessment.

A best practice of mobile app testing strategy strikes a balance between automated tests and interactive human-based testing (Figure 1). The ideal quality cycle begins with a battery of automated tests against the output of the continuous integration build process for your mobile app. After initial automated tests have verified that the latest build meets minimum quality criteria, the build can be distributed to testers and internal evaluators for interactive testing.

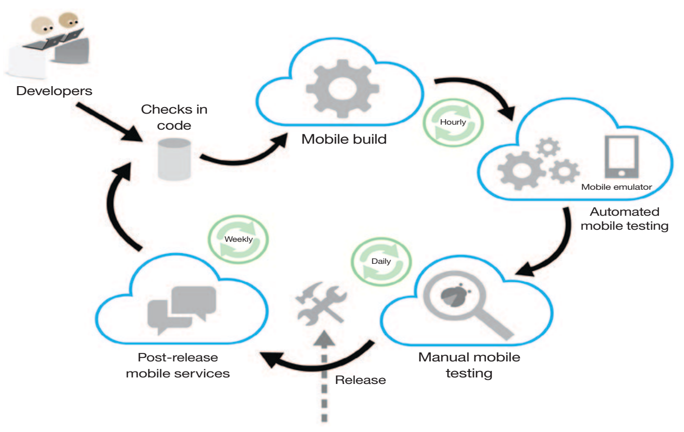

Figure 1. Ideal mobile app quality cycle

When the mobile app has passed all the tests, it is a candidate for release to production and distribution to real users. Even after the mobile app is in production, you can continue to obtain quality assessment data about the application.

Preproduction versus post-release quality assessment

The main focus of quality assurance for mobile apps is in the preproduction phases of the development lifecycle. However, quality assessment of the app should not end when the app is in production. Very important data about the behavior of the mobile app “in the wild” should be used to help the developers continuously improve the app.

Expecting every conceivable defect in a mobile app to be caught and fixed before it is released into production is not practical. Even when they are in the app store and installed on mobile devices, the best mobile apps continue to capture context information for every crash that occurs. They then deliver that information back to the app development team.

Many times, a crash only occurs under very specific conditions that are difficult or impossible to recreate in the test lab. The problem could arise from the use of an unusual or old mobile device model or from special network situations or the combination of certain other mobile apps running concurrently with the mobile app in question. These real-world circumstances are impossible to anticipate and impractical to cover in all permutations in preproduction test time frames. Therefore, your development organization has to depend on receiving good technical contextual data when they occur in the field, so the root cause of the crash can be quickly determined.

Besides capturing outright crashes, soliciting feedback from your app users about how they perceive the mobile app is also valuable. Most popular mobile apps include some kind of "in-app feedback" mechanism so users can send the development team a short message about how they view the app. The context of such feedback at the time it is submitted is very important because it helps the development team determine if special conditions are contributing to the user feedback.

Automated mobile app testing considerations

Several aspects of automated mobile app testing bear special attention. These include:

- The devices used for testing
- Isolation of the code that is running on the mobile device
- The specific technique employed to create and execute mobile test automation

Test devices

A crucial consideration for automated mobile testing, no matter the type, is which mobile device (or devices) to use. The automated tests could be executed on a small number of tethered real physical mobile devices. Or they could be run on emulator programs on the developer's workstation. Several vendors offer a remote "mobile device cloud" as a potential target for tests. Device emulator programs could even be scaled in a virtual device cloud that provides elastic computing capabilities as testing load varies between peak activity times.

Emulators and simulators

Emulators are included in all the native mobile operating system development kits, and simulators are available from several sources (such as IBM). Emulators attempt to replicate the actual mobile operating system on other hardware, such as a workstation. Simulators do not attempt to replicate the mobile operating system, but instead provide a lightweight simulation of the user interface only.

Using emulators and simulators for some amount of testing can be cost effective, especially in the early stages of code development. The typical code-deploy-debug cycle, especially for simulators, is much more rapid than for physical devices. Using these tools can eliminate the need for the real physical device. However, subtle differences in behavior exist between device emulators and real physical devices even for the very best emulator programs. In addition, simulators do not allow execution of some parts of the application logic flow (only the user interface look and flow). So, although emulators and simulators can be used to cut costs and speed development, they generally should not be the *only* form of test execution for mobile apps.

IBM offers mobile simulators as part of the development tools for the IBM mobile enterprise solution. Emulators are provided directly by the supplier of the mobile operating systems.

Device clouds

Broad-scale testing on real mobile devices is crucial for any app that will be released into the consumer market or used in organizations that have a bring-your-own-device strategy. Approaches for on-device testing include the category called device clouds. What about the problem of the sheer number of different physical mobile devices on the market? Thousands of different device types run different release levels of mobile operating systems and are connected to different network carrier providers and wireless networks. The combinatorial complexity of the universe of possible permutations is almost beyond comprehension. The cost of owning, setting up and managing all of those different combinations is completely prohibitive even for very well-funded projects.

A technique for addressing this problem is the employment of a "device cloud" testing solution. Device cloud describes a very large array of real physical devices that have been made remotely available for access on the internet much like general processing resources in a generic software "test cloud" solution.

The test organization arranges to reserve some mobile devices for test for a certain amount of time. The organization then deploys the mobile app code to the devices with whatever "on device" automated test solution is the choice of the test organization. After the testing cycle is complete, the reserved devices are relinquished to the "device cloud" and can be used for other mobile app testing.

This technique does not eliminate the need for manual interactive testing by humans, and in fact works best in conjunction with some form of execution of those other techniques. However, this method reduces the cost of ownership of the huge variety of device types that can be expected to be employed by the users of the mobile app after it is in production.

A test organization can invest in purchasing just a few key mobile devices and "rent" the rest of the combinations from the device cloud. The same automated mobile tests can be used on the standalone physical devices and the devices in the cloud so results are consistent. Although it handles the array of devices to test, this method does not provide for testing connectivity states other than turning off the connection. One of the most overlooked testing scenarios is what happens if the device disconnects during various key uses of the app. Some services do offer the ability to select which carrier the device is running on, but ideally tests should be run as close to real scenarios as possible.

Issues related to this kind of testing can arise, whether the resources in the cloud are general compute resources or mobile device resources. Security of the app being tested, the option of public or private device cloud and balancing the cost of the cloud with the potential cost of the defects eliminated are among these issues that any cloud testing solution has to address.

Crowd-sourced testing

Many organizations find it challenging to get feedback from their own internal users for prereleases of their mobile applications. As a result, a set of companies offers solutions that allow for distribution of an application to live testers out in the field.

What's interesting about these testing services is that you will get real user behavioral feedback. You can also test in regions where you will deploy. Users can even run through use scripts to ensure anticipated use is specifically tested, although caution is advised with this process because the results are not always reliable.

Crowd-source testing introduces the same security risks mentioned for the device clouds. If the application is accessing sensitive production systems or data that should not be public, this approach should not be considered.

Using service virtualization to isolate mobile code

Because mobile applications are multitier architectures, setting up the infrastructure to support testing of the code on the mobile device can be time-consuming and costly. All middleware servers and services must be running and available in the test environment because using real production servers for testing purposes is generally not advisable.

In addition, test case failures can occur not because of defects in the code, but because of problems in the connected application components that run in other tiers. In other words, if the middle-tier app server has a problem, the mobile device that is accessing it will likely fail its test case.

Cost and deployment delays can be eliminated with solutions that effectively replicate connecting components of the multitier system. The testing then can be concentrated narrowly on the code executing on one specific tier of the app. By using solutions such as the service virtualization capability in IBM® Rational® Test Workbench, test teams do not have to set up complex middleware test environments in order to test code that runs on the mobile devices.

Service virtualization can emulate the middle-tier and back-end services and protocols so that the testing focuses on the client tier of the mobile app. The middle-tier components of the mobile app must be validated also. Service virtualization can emulate the protocols delivered by the mobile device clients and tests can focus solely on the middle-tier functions and services.

Mobile test automation techniques

Staying ahead with mobile apps means frequent iterations with new features. As companies create more updates for their apps, testing quickly gets out of control if everything is being done manually.

Ultimately, augmenting and accelerating manual functional verification with some form of automated testing is desired. This area of the software testing market is new and evolving. Various solutions and methods have been created by different vendors. Some are better suited for typical mobile business applications than others.

Mobile app programmatic instrumentation

Programmatic instrumentation typically places some kind of additional code on the device where the automated testing is to occur. This code acts as a local “on device” agent that drives automated user input into the application and monitors the behavior of the application resulting from this input.

The instructions for telling the agent what to input into the app are typically formatted as either a script or an actual computer program (for example, written in Java). Creation of these automated test instructions usually requires some proficiency in programming, and many test organizations are short on such skills. Furthermore, the creation of automated tests is a development effort in its own right, which can delay the delivery of the mobile app into production.

IBM’s point of view is that creating automated mobile function test scripts should be possible for testers who have no programming skills at all. The tester should be able to put the application into “record mode” and interact with the app normally. The testing solution (that is, the test agent running on the device) should then capture the user input from the tester and convert it into a high level automation script. After the tests have been “recorded” into a language similar to natural written instructions, these test scripts can be further edited, organized, managed and replayed whenever necessary.

Furthermore, because the language is nearly natural human language, the tester can easily read and modify the script to add elaboration and additional verification points to the instructions. If the language is suitably abstracted from the details of the mobile operating system, these scripts can be used on real physical devices other than those that captured the script in the first place.

IBM offers just such an automated mobile app test solution and delivers it in both the mobile app development environment (IBM Worklight® Studio) and a software testers' solution (IBM Rational Test Workbench). This solution for creating test automation is quite complementary to other mobile app testing techniques and can be very effectively used in combination with them.

Random generated mobile tests

Random generated tests have the advantage of not requiring any scripting or coding of automation instructions. This type of testing introspects the mobile app and generates random pathways of interaction with the app. The random input to the app is executed until a fatal error in the app occurs (for example, a crash or freeze of the app).

Randomly generated automated tests have proven to be quite valuable for quickly uncovering serious defects in mobile apps that typical scripted tests are likely to miss. Any battery of automated tests for a mobile app should include some amount of this kind of testing. It has been known to uncover defects in the app within just the first few seconds of execution. As the defects in the mobile app surface and are fixed, it might take more time to identify problems in the app using random tests. However, that should be considered a good outcome from use of this testing method.

Image recognition automated mobile tests

Another form of automated testing for mobile apps employs the display images from the mobile device and pixel locations on the device screen. The device display image can usually be captured and automatically compared to a known good image for verification. Automated input to the app being tested is defined as a set of tap events targeted at a pixel location on the device display rather than at an internal programmatic application object.

The advantage of this form of test automation is that it is platform-neutral so the mobile operating system and the technology used for internal implementation of the app are of no consequence. It is more similar to how a real human interacts with and perceives the mobile app. And, highly skilled programmers are not required to produce the automation scripts.

The downside is that the scripts are highly susceptible to display changes in the mobile app. If the location of a particular widget is changed by a new build of the app, the script that depends on the pixel location of that widget will be broken. Some vendors employ advanced algorithms that reduce this "brittleness" in the test scripts. But this technique is so sensitive to app display changes, that it is best used later in the development process when the amount of anticipated visual changes in the app is minimized.

Making manual testing more effective

Manual testing is the most common method of mobile testing in use in the industry today. It is an essential element of any quality plan for mobile apps because it is the only technique that currently provides results for how consumable the app is. The success of mobile apps depends on how consumable and intuitive they are. So far, a mechanism for automating the testing of this aspect of code is not available.

Manual testing is also the most time-consuming, error-prone and costly technique for mobile testing. It can be combined with crowd-sourcing and "device-cloud" techniques so the costs and time required can be mitigated somewhat. Solutions that organize the manual test cases, guide the tester through execution and store the test results can substantially reduce the costs associated with manual testing.

IBM offers a hosted, software-as-a-service capability designed to make interactive manual testing significantly more efficient and effective. IBM Mobile Quality Assurance services begin the process with an "over-the-air" app distribution capability. This capability makes it easier for app developers to deliver updates and new mobile app builds to a targeted set of testers directly on their mobile devices.

"In-app" bug reporting and defect troubleshooting

When a new build of the app is ready (that is, it passes the initial automated tests), the developer can upload the app binary (.apk or .ipa file) to the IBM Mobile Quality Assurance service. The developer can also identify the people who should be notified about the availability of this new build, and they receive an email about the new build. When the tester clicks a link in the notification email, the new app build is automatically installed on their mobile device, ready for immediate testing.

Mobile app testers can be confident that they have the correct build of the app to be tested. During their interactive manual testing, when they encounter a defect, they can use IBM Mobile Quality Assurance "in-app" bug reporting to submit it right from their mobile devices. The testers simply shake their mobile device and the app being tested will go into "bug reporting mode." This mode suspends the normal behavior of the app so the user can capture one or more screen images from the mobile app. The screen image can be augmented with annotations made with the tester's fingers (lines, circles, arrows and more).

After the screen image is captured, the tester is presented with a text box to be used to describe the defect in words. After the description is entered, the tester taps the "Report" button and the defect information is sent to the IBM Mobile Quality Assurance service. Rich technical details of the context of the mobile app and the device on which it was running are also captured and sent. This rich context includes:

- Mobile device type
- Mobile operating system and release level
- Network in use, including carrier or wireless details
- Memory available and in use on the device
- Logging output up to the point when the defect is reported
- Battery level

This detailed technical information is invaluable in helping the mobile app developers troubleshoot the defect and understand the root cause of the problem.

If you use IBM Bluemix DevOps services, you can configure IBM Mobile Quality Assurance services to automatically open work items for each crash or bug report.

Crash data capture and analysis

IBM Mobile Quality Assurance services is designed to capture every application crash, whether it occurs during preproduction testing or after the app is in production. Each time the application crashes, its entire context and details about the device it is running on are sent to the IBM Mobile Quality Assurance services. IBM Mobile Quality Assurance services analyzes this "must-gather" data and makes it available to the development team. These capabilities can be used during automated testing, during manual, interactive testing and even after the app has been released to the app store.

The service can also recognize and aggregate crashes that occur at the same spot in the mobile app. With these capabilities, you can see how many times a crash occurs at the same location in the app logic. This important information helps your development team prioritize which crashes to fix. A crash that occurs 1000 times most likely should be fixed before one that occurs only once or twice.

Performance testing

Performance testing can be viewed from two distinct directions when it comes to mobile apps. One form of performance testing is to systematically increase the number of concurrent mobile client instances and drive large loads of requests to the middleware and server components of the app. This load and stress testing is important because it helps ensure that the services that support the mobile app can absorb the high traffic associated with mobile apps.

The other dimension of mobile performance testing measures how efficiently the code makes use of the device resources. A mobile app can inadvertently consume too much of the device's resources (memory, cpu, network, battery). Such a "resource hog" is likely to be perceived to have poor quality and to be deleted from the user's phone.

Load and stress performance testing

Performance testing for load and stress is one of the more difficult testing scenarios. Typically, this involves setting up a test harness that pours sample transactions to your backend at whatever rate you set. For internal enterprise apps, test peaks and maximums are critical; consumer apps require those tests, plus spikes and lulls to replicate the natural use of the app.

Load and stress testing products such as IBM Rational Performance Tester (part of the IBM Rational Test Workbench) include a recording proxy for the traffic patterns between your mobile app and the services it calls over the network. This solution stores the recorded interactions as a script. The script can then be run in hundreds or thousands of virtual clients to apply load to the mobile app back-end services. A good practice is to concurrently run functional tests against the mobile app while the mobile app back-end services are under this synthetic load. This practice enables you to determine if the behavior of the client side changes when the back-end services are under a heavy load.

IBM's point of view on performance is that direct integration architectures are brittle. Generally, they do not handle issues that are related to volume and stress gracefully. Mobile middleware, however, creates a separate tier that can shield the back ends from catastrophic conditions.

Mobile client resource metrics

Client-side performance is affected by many different aspects, from the level of graphics used in an app to poor coding and bad practices. By measuring and monitoring battery use, device resource use (for example, getting GPS coordinates), and screen loading time, you can tune your apps to remove potential resource hogging.

Correlating the measurements of mobile device resources with the functional tasks in the mobile app is especially effective. Being able to associate a spike in memory usage with a specific app functional activity enables developers to quickly pinpoint what area to address to resolve excessive resource usage.

User sentiment as a measure of quality

After your app has been released into production, users can post reviews in the app store. App store review comments offer a rich source of quality assessment.

If your mobile app has only a few reviews in the app store, you can read each one and gain insight into the sentiment of the users. But after the number reaches several dozen or more, an analysis tool is required for effectively and efficiently mining the key insights from that amount of data. Such sentiment analysis can shed light on trends in the perception of the audience for your mobile app. You can also correlate this insight with the quality assessment measurements such as the crash data from production versions of the app.

For example, the IBM Mobile Quality Assurance service includes an app store review analysis capability. It gathers all of the review text and searches for a set of special "user sentiment" key words. Analysis for app store reviews is organized into 10 distinct "attributes" about your mobile app, such as:

- Usability
- Stability
- Performance
- Elegance

For each of the user sentiment attributes for your mobile app, you can see the analysis that produced the score for that attribute. You can even see the specific list of reviews with comments about that attribute. Such sentiment analysis is very useful when combined with the hard technical evidence in the crash reports and in-app user feedback records.

Some users do not post reviews in the app store, but instead share their perception of your mobile app in their various social media networks such as Facebook, Twitter or LinkedIn. Social media analytics can uncover information about what your users are saying to their community about your mobile app.

A full spectrum of mobile app quality: IBM's recommendations

IBM's point-of-view for a comprehensive approach to mobile testing and quality management is that it should encompass the full spectrum of mobile test techniques currently in use in the industry. The following elements should be part of this solution.

Selective, targeted test device usage

IBM's point of view is that you should concentrate your investment in real physical devices on only the highest priority device types and OS release levels. Then, rent the other permutations from a device cloud vendor that is integrated with your test management solution, such as IBM Rational Quality Manager. You can apply the same tests to the "cloud devices" that are used for your in-house physical devices for greater consistency in your test results.

A thorough automated testing strategy

Running a suite of automated mobile tests against each build of your mobile app is critical, as is automating the tests for the code that executes directly on the mobile device. IBM recommends using the IBM Rational Test Workbench automated mobile testing capability (either directly or with the IBM Worklight Studio development environment). Hiring specialized skilled programmers to produce the automated versions of your mobile test cases should not be necessary.

In addition, you should use emulators as target test devices for your everyday automated regression testing. Use the same automated mobile test capability to execute emulator test cases that you use for the real physical devices. An automated mobile app testing solution, such as IBM Rational Test Workbench, should work on both emulators and real devices.

No thorough automated mobile testing strategy is complete without service virtualization, such as that available in IBM Rational Test Workbench. Use service virtualization to:

- Isolate various tiers of the mobile app so that testing can be concentrated solely on those specific tiers.
- Test the code on your mobile device without setting up and running a complete middle-tier.
- Test the middle-tier mobile app logic without coordinating large numbers of mobile device clients.

Tools that optimize manual interactive testing

After sufficient automated testing has completed successfully, distribute the app to a group of human testers to perform interactive manual testing on the app. Use “in-app” user feedback to make it easy and efficient for users to communicate with your development team about their perception of the mobile app. Capture the context of the app and the device on which it is running every time a user submits feedback.

IBM recommends instrumenting your mobile app with IBM Mobile Quality Assurance code so your manual interactive testers can submit bug reports directly from the app being tested on their mobile devices. Employ IBM Mobile Quality Assurance crash data capture services to obtain deep technical context information about each crash, whether during preproduction testing or after the mobile app is in production.

Organization, management and analysis

Testing mobile apps can be complicated with many piece parts and results to monitor. IBM highly recommends organizing and managing the test tools for mobile app testing for mobile front ends, supporting middleware and back-end services. A test management product such as IBM Rational Quality Manager can help. You can use it to consolidate and link the test results from multiple tools into a single mobile app quality metric, and link data from test case failures back to development work items for defect removal. You can also use it to organize your manual test cases into logical suites, thereby potentially reducing the costs and increasing the efficiency of your manual test efforts.

In addition, IBM recommends that you use an analysis tool to aggregate and gain insight from the app store review comments for your app. Correlate this app store user sentiment analysis with the other quality assessment data that you continue to gather about your mobile app in the field.

Conclusion

Mobile apps are becoming the chief way that users interact with their organizations, use data and work. Therefore, your mobile apps must be top-notch, easy-to-use, intuitive, high-quality, and high performing. Mobile app testing helps ensure that you are delivering on all these requirements. Therefore, testing all aspects of the software, including cloud-based and back-end system services, is critical to the success of a mobile app. Engaging with users is also crucial to success and being able to rapidly gather and respond to feedback makes the difference between a good and bad app rating.

IBM understands mobile app testing and quality measurement. As a result, IBM’s point of view of mobile app testing is that it should be extensive and thorough, with a combination of automation, manual interactive testing backed by delivery tools and defect tracking, sentiment analysis and test organization and management. IBM’s tools for mobile testing, therefore, are designed to enable thorough and comprehensive testing. These solutions can enable you to test quality mobile apps for function, performance and user acceptance. You can gather quality metrics and supporting data to determine the actual success of a mobile app. And, you can test in the very rapid time-to-market paradigm required by mobile.

For more information

To learn more about IBM solutions for mobile app testing, please contact your IBM representative or IBM Business Partner, or visit the following website: **ibm.com**/rational

Additionally, IBM Global Financing can help you acquire the software capabilities that your business needs in the most cost-effective and strategic way possible. We'll partner with credit-qualified clients to customize a financing solution to suit your business and development goals, enable effective cash management, and improve your total cost of ownership. Fund your critical IT investment and propel your business forward with IBM Global Financing. For more information, visit: **ibm.com**/financing

RAW14356-USEN-00