A mobile application development primer

A guide for enterprise teams working on mobile application projects

IBM

Executive summary

Industries of all varieties have begun to realize that the target audiences for their business applications have shifted in massive numbers from the use of traditional personal computers, such as desktops and laptops, to using mobile devices such as smart phones and tablets for accessing the internet and for obtaining the information they seek. This applies if the intended audience for the application is a direct customer of the enterprise (Business-to-Consumer apps, or "B2C"), or if the targeted user is an employee or business partner ("B2E" and "B2B", or Business-to-Employee and Business-to-Business apps). Across the globe, more people are now using mobile devices that they can carry with them wherever they go, and which are more user-friendly and intuitive to use, as their primary means of obtaining information and requesting services over the internet.

This crucial shift in end user behavior has motivated enterprises to develop mobile channels for their existing business applications, and to plan for new kinds of applications that can exploit the unique characteristics of the mobile devices on the market. As with all major evolutions in the information technology industry, the first years of this shift have seen frantic activity to meet demand and create market presence without considering more strategic issues such as application development costs, maintainability, quality and security. As the mobile application market matures and the initial rush to market settles, these more comprehensive software development issues are being brought into focus by those in the enterprise responsible for longer-term planning and economics.

IBM has established a reputation as a prudent and responsible software development partner for enterprises in many industries. A recent paper from IBM Global Services, titled "Establishing an effective application strategy for your mobile enterprise,"[1] provides a broad overview of IBM's recommendations for planning, developing, deploying and managing mobile applications. This paper covers one topic of that broader paper and focuses on a comprehensive approach for development of mobile business applications. The technique described combines best practices for collaborative software lifecycle management with newer requirements unique to the creation of mobile applications. The content of this paper is intended to provide value for all of the roles involved in mobile enterprise business application development projects: architects planning for mobile projects; development teams making implementation decisions; project managers establishing the details of the project activities; test organizations addressing these new applications; and executives that need an understanding of how these new mobile apps fit with the existing enterprise applications and development processes.

Unique challenges for mobile application development

The creation of applications intended to execute on newer mobile devices such as smart phones and tablets involves unique requirements and challenges.

Form factors and user input technology

The first and most obvious unique aspect of mobile applications is that the form factor for display and user interaction is significantly different from prior forms of software. Smart phones usually provide only a four-inch area in which to display the application content and offer lower screen resolution pixel density compared to personal computer (PC) displays, which are trending toward greater display sizes and number of screen pixels. Even tablet devices have generally lower display sizes than PCs, especially when compared to the large flat-screen displays in use for newer desktop PCs.

A smaller form factor means that the amount of data displayed to the end user, and layout of that data, needs to be different for these applications than for apps expected to run on PC devices. Significantly less data can be displayed at one time and therefore it must be exactly the "right" data, most relevant to what the user needs at that point in the application.

Another obvious physical difference for mobile applications is that the mechanisms for user input are different. Mobile devices have pioneered the use of non-keyboard "gestures" as an effective and popular method of user input. Touch, swipe, and pinch gestures must be planned for and supported in a satisfying mobile application user experience. These tactile end user input mechanisms have proven to be so popular that they are now being retrofitted into traditional desktop PC systems such as the Apple "Lion" OS X release and Windows 8 "Metro" OS. In addition to tactile user input, mobile devices are a natural target for voice-based user input. In fact, the traditional keyboard typing form of user input is probably the least effective and least popular mechanism for input delivery from mobile application users.

Besides input directly from the end user, mobile devices have the capability to receive input from other sources, such as geo-location input from the GPS component of the device and image information from the camera typically built into the device. These unique forms of input must be considered during mobile application design and development. They offer new and valuable mechanisms to make mobile apps more powerful and useful than applications with a more limited array of input possibilities.

Usability and user interaction design

Several factors motivate the need for more attention to usability and user interaction design for mobile applications. One is the difference in form factors and user input methods. It is much more difficult and time consuming to plan how to display only the data that is precisely necessary than it is to simply display all possible data and let the end users visually sift through it for what they want. An analogy for the written word is that it is harder to write a concise abstract than it is to write an entire paper. The mobile app designer has to consider the screen real estate. When an application needs to present a broader scope of data, with multiple layers of detail, it is usually better to use a progressive discovery approach that allows the user to "drill down" into incrementally greater levels of detail that is focused on fewer specific items.

The rich variety of input methods available on mobile devices also is a motivation for early design work to identify and use more efficient ways to deliver input data than the simple "just type it in a form" design that is a default for traditional web and PC applications. Extensive keyboard typing for mobile apps must be avoided in order to reduce end user frustration, particularly with drastically smaller touch keyboards and lack of traditional typing feedback. Identifying non-keyboard ways in which information can be gathered and delivered to the mobile app is a significant design challenge.

There is yet another more subtle motivation for extra attention to the design effort for mobile applications. The way in which end users interact with mobile devices and the applications running on them is different from how they interact with stationary PCs and even laptops. End users of a mobile device are typically holding the device in their hand while also interacting with the surrounding reality of their physical situation. These application users typically cannot concentrate intently on the mobile app for very long before they need to switch their attention to their physical surroundings. The interaction model for users of mobile apps is short, interrupted, and "bursty," meaning that they need to very quickly complete the application task before switching attention.

All of these factors drive the need for more investment in user-centered design for mobile applications very early in the development project. Ideally, these usability considerations and design aspects should be codified in the requirements for the mobile application and then linked to the later stage development deliverables, along with the tests that validate that the user interaction and "consumeability" of the app is as satisfying as possible.

Choice of implementation technology

There is a spectrum of implementation choices for mobile applications in the market. There is no one perfect answer for the choice of implementation for a mobile application, and all of the choices across the spectrum have their advantages and disadvantages. Therefore, the challenge for mobile development teams is to understand the trade-offs between the technologies and make a choice based on the specific application requirements. The previously-referenced IBM Global Services paper includes a concise description of the implementation choices, along with a comparison table. This paper provides a few supplemental considerations.

The choice of implementation technology for a mobile project will have an impact on other decisions related to the application's development. It may limit the choices for development tools. The implementation choice will likely have an impact on the team roles and structure. It may have an impact on how the application is tested and verified, and how it is distributed and delivered to the end user. So, the choice of implementation approach for a mobile application is a crucial, early-stage decision to be made very carefully.

Native application implementation

A "native" implementation means that you are writing the application using the programming language and programmatic interfaces exposed by the mobile operating system of a specific type of device. For example, a native implementation for an iPhone will be written using the Objective-C language and the iOS operating system Application Programming Interfaces (APIs) that Apple supplies and supports.

Native application implementation has the advantage of offering the highest fidelity with the mobile device. Since the APIs used are at a low level and are specific to the device for which the application is dedicated, the application can take full advantage of every feature and service exposed by that device.

Native implementations of mobile apps are completely non-portable to any other mobile operating system. A native Apple iOS app must be totally rewritten if it is to run on an Android device. That makes this choice a very costly way of implementing a mobile business application.

Web applications

Newer smart phones and tablets come with advanced web browsers pre-installed, and it is very feasible to implement a mobile business application that is a standard web application, plus special style sheets to accommodate the mobile form factor and approximate the mobile device "look and feel." Mobile applications implemented using this approach support the widest variety of mobile devices, since web browser support for JavaScript and HTML5 is fairly consistent. There are several commercial and open source libraries of Web 2.0 widgets that help with this approach. The web programming model for mobile application implementation also has an advantage for enterprises that already have developers trained in the languages and techniques for web application development.

The disadvantage of pure web application implementation is that such apps have no access to functions and features that run directly on the mobile device, such as the camera, contact list,

and so forth. However, if your mobile application does not depend on local services running on the device, the pure web application approach could be sufficient. As the HTML5 specification matures and becomes more widely supported by the mobile web browsers, many of the services local to the mobile devices will become exposed for pure web applications through that W3C programming standard.

Another consideration that differs between web applications and native applications is the manner in which the application is distributed and made available on the device. Native applications must be downloaded and installed from some kind of "App Store," such as the publicly accessible Apple iTunes store or Google's Android Marketplace. The app store distribution mechanism has the advantage of allowing the mobile app to easily be located using search algorithms. Enterprises sometimes appreciate the market visibility and end user feedback that mobile app stores facilitate.

The downside of app stores, especially the public ones, is that they sit directly between the enterprise and its intended target audience. All mobile application updates must go through the app store, and it can be difficult to remotely control and manage the mobile app that has been delivered through app store mechanisms. Web applications aren't distributed through an app store. The end user simply enters the web address for the application into the mobile web browser, and the application is delivered over the Internet. Updating the mobile web application is as easy as updating the server or servers that host the app. The next time any user accesses the web site, the new version of the mobile app is downloaded to the device.

Hybrid mobile application implementation

Hybrid mobile application implementation is a form of compromise between pure native implementation and pure web implementation.

You write the mobile apps using industry standard web programming languages and techniques such as HTML5 and JavaScript. But, you package the app into a natively installable format that is distributed through the app store mechanism.

Hybrid apps are linked to additional native libraries that allow the app to have access to native device features from the single application code base. Because the bulk of a hybrid application is implemented using technology not unique to any single device, most of the code for the application is portable and reusable across many different mobile operating systems. However, small segments of native code also can be integrated with the hybrid app. So the developer can decide how much of the application implementation is a shared, common code base and how much is device-specific customization.

You can also choose how much of the code to package as a "native" installable app delivered through the app store and how much to download over the network. The first elements of the app to be displayed can be packaged for installation directly on the device, so they load quickly when the user launches the app. Other, more dynamic elements can be structured as web pages that are managed on a server and always provide the latest version of the application when accessed.

For the average mobile business application, many industry analysts have a strong conviction that the economics of code reuse and flexible application development will favor the compromise hybrid approach over the long term.

Mobile application build and delivery

Because of the strong business motivations to deliver mobile applications into the market quickly, mobile development projects typically have extremely aggressive time lines. Inception-to-delivery periods of a few months are common. The pressure to deliver mobile apps quickly results in the adoption of agile development methods for most mobile projects.

An important element in agile development practices is continuous integration and builds. Application changes delivered by developers need to be processed immediately for all of the mobile operating systems on which the application is required to execute. If the mobile application is a hybrid or native implementation, several different builds of the application need to be triggered each time a change set for the application is delivered by a developer. The build setup and configuration for each supported mobile environment will be different from the others, and it is most likely that a small "farm" of build servers will need to be provisioned and available to handle these builds of the mobile application for multiple operating systems.

Testing

Another area where mobile application development poses a huge challenge is testing. Testing for mobile applications represents a quantum leap in complexity and cost over more traditional applications. Unlike traditional PC and web applications, the range of potentially supported mobile devices and release levels is staggering. It is quite common to see test matrices for mobile projects that contain hundreds, and even thousands, of permutations of device, mobile OS level, network carrier, locale, and device orientation combinations.

There are more variables thrown into the equation for mobile testing that aren't relevant for other kinds of software. The same model of device may function in a subtly different way when connected to a different carrier network. Also, the quality of the network connection can have a profound impact on the behavior of a mobile application. Even the movement of the mobile device itself may be an important factor in the behavior of the application since some applications specifically exploit device movement.

The majority of mobile apps have a multi-tier architecture, with the code running on the device itself the "front-end" client to data, and services supplied by more traditional middle-tier and data center "back-ends." Effective and comprehensive testing of mobile apps requires addressing all tiers of the application, not only the code on the mobile device. The setup and availability of test versions of the middle tier and back-end services can present very large cost and complexity challenges for mobile applications testing.

Many mobile projects start by using manual testing approaches. This is the most obvious way to begin testing quickly. You would have to buy all of the various mobile devices that you plan to support with the app, and pay someone, or more likely a team of people, to tediously go through a written script of instructions describing the tests on every one of those devices for every build of the application. Such manual testing is extremely expensive and inefficient. Nevertheless, manual testing does serve an important purpose by providing a mechanism for obtaining crucial usability feedback for the app.

Instead of buying the real mobile devices, you could rely on mobile device simulators and emulators for your testing. Using this approach, a software program running on a desktop workstation takes the place of a real device. The use of simulators and emulators for mobile application testing can be valuable for tasks such as developer unit testing. Some of the device emulators are excellent, but some are not that good at replicating the real device. Therefore, in either manual or automated testing, some form of testing on the real mobile device is always essential.

There are mobile app testing solutions that rely on running an agent program on the device that a test script can interact with in an automated execution. This approach has the flexibility of

using either real physical devices or emulators for testing, with the added efficiency of automation. However, the test organization bears the costs of setting up the devices to be tested and installing the test agent on them.

Another approach to address the mobile app testing challenge is to make use of what can be called a "device cloud." A cloud can expose resources that are actually mobile devices instead of general-purpose computers. Instead of "renting" a Linux virtual machine for a few hours or days of testing, you can rent a specific model and release of a mobile device. This approach saves the enterprise the costs of purchasing maybe hundreds of devices and managing all of them for testing.

How is mobile development similar to other software development?

Even though there are unique aspects to mobile application development, many of the roles and tasks involved in the overall development lifecycle are the same as for enterprise-class development of other kinds of software. In the paper "Measuring Agility and Architectural Integrity,"[2] Walker Royce describes the key techniques and practices for effectively delivering software using agile and test-first principles. The software delivery practices in Walker's paper are a perfect fit for mobile development projects.

Full lifecycle for the project

The lifecycle of a software development project generally follows a similar pattern, regardless of the type of software being created. It starts with the business decision, based on some analysis, to invest in the application. Requirements for the application are captured and elaborated. These application requirements are further decomposed into user stories and feature work items, which are assembled into a plan of work for the iterations and releases to be completed for delivery of the application. Team members acting in various roles are assigned the work items and use various tools to complete the work and deliver whatever that work result consists of into the project. The resulting application is tested and certified to deliver the requirements. The exact process and lifecycle followed for software projects at a particular company usually is tailored to that specific enterprise's goals and policies.

This lifecycle is the same for mobile applications. Mobile application development is generally characterized by small teams, use of existing infrastructure, and highly user-interactive applications. Agile methods and test-first principles are ideally suited for such a scenario. Though the specific requirements for a mobile app development are likely different from some other software development, the tools and processes for gathering, elaborating and communicating those requirements are the same. The need to link the requirements to the code changes that deliver the implementation of those requirements also is the same for mobile applications as it is for other software. In other words, the flow of the project and the need for integration and traceability across the project is the same for mobile and other software development projects.

Integration of multiple tools

There are very few, if any, software development projects that can be delivered using one single development tool. Most projects involve a wide range of tools from different vendors, designed to meet the needs of a specific role or task in the overall lifecycle of the project.

For example, an individual developer of code for a mobile application may find that a simple code construction tool, matched to the mobile platform on which the application is to execute, will suffice for his or her needs. However, that tool is

missing features that facilitate the collaboration and coordination needed when an agile development team is involved in creating the application.

By integrating the individual developer's code construction tool with a compatible collaborative team development platform, agile teams can achieve improved efficiencies and quality.

Need for collaboration across the team

Mobile applications are typically created by a small team with varying skills and expertise. A typical team may consist of a couple of developers of the fundamental business logic and web services, a couple of user interface (UI) developers, a user experience designer, a few testers, and a team leader or manager.

Given the typically aggressive time frames for delivering mobile apps, even a small team must operate at peak efficiency. Any delay due to misunderstanding or miscommunication between team members can throw off the entire delivery schedule.

Mobile application projects supported across different mobile operating systems require code sharing and reuse. One developer may specialize in Android and another in iOS skills. Clear understanding of the work that team members are expected to perform, and when they need to deliver it, is essential. Project requirements, timelines, plans and so on are shared in that case whereas only source code, tests, and builds may differ.

Integrated change management, software version control

All code changes associated with a particular work item need to be tied together into a specific "change set," or list of changed source code files, that is delivered in one shot so that the full code change can be tracked as a unit. Ideally, this process of assembling a change set should be as unobtrusive and seamless as possible, so it does not cause interruption of the developers' concentration on the logic they are creating.

The processes for version control and merge/rollback also need to be automatic and intuitive. Any time a developer has to switch their working context in order to perform some kind of task, it represents a point of interruption and a potential "speed bump" in the development process.

Need for traceability across the project

The typical agile team approach to software development is to define multiple short iterations in which a small set of application enhancements are to be implemented and validated. A typical agile iteration is from two to four weeks long. The team leader can work with the team to map work items from a backlog list into the specific iterations and assign the work items to individual developers.

As the developers pick up the work items and begin to make progress on them, their effort needs to be automatically recorded and made available to the team leader to track and view. This makes the information about what has been completed, what is being worked on at the moment, and what is still to be done, easy to track and view in a dashboard presentation. Everyone on the team needs to be able to see how the iteration is progressing and the status of the work items planned for that iteration.

When the testers on the team start the functional testing of the mobile application, they need to open work items in the shared development project for defects uncovered during the course of testing. If the test case that failed is linked to a particular change set or feature item in the project plan, then the information about the code that was changed, and is likely to be associated with the test case failure, can automatically be entered into the defect data. Furthermore, if the change set is linked to the original requirement that motivated the code change, there is traceability throughout the whole project lifecycle, from the original requirement to the test case that verified that the requirement was delivered in the application.

This kind of "whole project view" and end-to-end traceability is extremely important for any kind of software development project, but especially relevant to mobile application development teams working on tight schedules and employing agile development methods. These kinds of lean development teams cannot afford to spend time tracking down details about whether and when a particular requirement was verified and delivered.

Choosing a mobile application development solution

The IBM approach to mobile enterprise application development combines years of experience in the field of general enterprise software development processes with new tools and techniques that are specific to mobile devices and their underlying software foundations.

With extensive expertise in the design and deployment of enterprise software across a wide array of industries, IBM offers customized solutions for the development needs of mobile application projects. IBM itself has gone through a transformation to employ agile methods across thousands of projects involving tens of thousands of developers.

Collaborative Lifecycle Management (CLM) for full project visibility

Without integrations throughout the mobile application delivery lifecycle, development teams are left to operate in silos. When silos form, product delivery effectiveness suffers. In order to deliver compelling mobile enterprise application solutions that respond to changing market needs and standards, software engineering teams must perform efficiently and manage all of the lifecycle work products collaboratively.

The IBM Rational solution for mobile business application development offers an integrated lifecycle management platform that supports collaborative tasks and helps link the various artifacts developed over the course of the product lifecycle. This solution also enacts delivery workflows and provides task management capabilities to run the mobile application development project effectively. It is augmented with integration of mobile-specific capabilities in the phases of the project lifecycle where such capabilities are needed.

Agile team collaborative development tools such as IBM® Rational® Team Concert™ software enable the definition of multiple short iterations within the overall project. Work items can easily be moved from the project backlog to a particular iteration plan.

As the developers edit files inside mobile application code development tools integrated with the software version control system, a change set is automatically updated and maintained. The developers don't have to do anything to produce the change set other than edit the files they need to work on.

Change sets can be shared among members of the team before being fully integrated with the main code stream. Therefore, a change set created by the web service developer, altering the format of data supplied by the web service, can be shared with the UI developer working on the logic that displays the new data, without the rest of the team being affected. Once both UI code changes and web service code changes are matching and deemed ready, they can be integrated in one synchronized task into the mainline code stream for the rest of the team to pick up and use.

IBM Mobile Enterprise strategy delivers a mobile application runtime

The combination of a collaborative lifecycle management platform integrated with code development tools specifically targeted for mobile applications is an important component of the IBM comprehensive mobile application solution. However,

some of the challenges for enterprise-class mobile application development cannot be addressed by development tools and practices. In order to deliver the capability for a single, common mobile application programming model, we offer IBM Mobile Enterprise software runtimes.

The IBM mobile business application development solution combines powerful team development capabilities embodied in IBM Rational Collaborative Lifecycle Management with the mobile tools and runtime capabilities delivered in the IBM Mobile Enterprise solution.

A comprehensive testing approach

The implication of comprehensive, multi-tier mobile application testing is that more than one test execution capability must be used and coordinated into a single application quality result. IBM Rational Quality Manager software is an excellent choice for tying together and managing the various test execution engines that are required for mobile testing. Our test environment enables you to enact an integration testing first approach that can help eliminate the big issues earlier in the lifecycle and improve economic governance. Walker Royce's paper, "Measuring Agility and Architectural Integrity," describes in detail the technique for testing the hardest problems first.

As described in a previous section of this paper, there are several techniques for testing and validating mobile apps in use in the market today. Managers of an effective development project will employ all of the applicable techniques against its mobile application because each technique has its strengths and weaknesses. There is no single perfect answer for mobile app testing, and the various techniques available are not mutually exclusive. The most effective testing strategy balances the use of all forms of mobile test execution and compiles the results from each into a comprehensive overall mobile application "quality metric".

Examples of the techniques for mobile app testing supported by the IBM approach include:

Manual testing

Manual testing is the most common approach for mobile testing in use in the industry today. However, manual testing is also the most time-consuming, error-prone and costly technique for mobile testing. Solutions that organize the manual test cases, guide the tester through execution and store the test results can substantially reduce the costs. Rational Quality Manager software offers these capabilities.

Emulators and simulators

Emulators come with all of the native mobile operating system development kits. Simulators are available from several sources, including IBM, which offers mobile simulators as part of the development tools for the IBM Mobile Enterprise solution.

Protocol virtualization, application tier isolation

Because mobile applications have a multi-tier architecture, the process of setting up the infrastructure to support test execution of the code on the mobile device can be time-consuming and costly. Cost and deployment delays can be minimized by using the IBM Rational solution. Test teams can avoid the need to setup complex middleware environments in support of test execution for code running on the mobile devices. The Rational solution can emulate the middle tier and back-end services and protocols so the test execution can concentrate on the client tier of the mobile app that is running on the device itself.

On-device instrumentation and agents

Ultimately, there is a requirement to replace manual functional verification with some form of automated testing of the code that is executing on the mobile device. There are a variety of approaches that have been created by different vendors. A typical approach is to place 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. This technique for automating the mobile function tests is quite complementary to the other techniques described in this document, and can be used very effectively in combination with these other techniques.

Device clouds

The cost of owning and setting up and managing all of the different combinations of mobile test devices is prohibitive, even for very well-funded projects. A technique that can address this problem is to employ a "device cloud" testing solution. This approach is effective at reducing the cost of ownership for the huge variety of device types that exist and can be expected to be employed by the users of the mobile app once it gets into production. IBM offers integration between the overall mobile testing management solution, Rational Quality Manager software, and a variety of business partners who have device clouds.

Conclusion

As more and more enterprises in all industries realize the need for mobile versions of their business applications, there is a need for an enterprise-class approach to mobile app development. IBM has established such an approach.

The IBM approach to mobile application development emphasizes five key themes:

- Simplify the mobile app user experience.
- Integrate first for improved economic governance of the mobile app project.
- Ensure traceability of requirements to tests that verify those requirements.
- Enact ultra-agile methods.
- Evolve automated regression test suites for rapid deployment and reduced cost-of-change.

This approach enables mobile specific tools and technology to be used with the same efficiency and rigor as other kinds of enterprise application development.

About the author

Leigh Williamson is an IBM Distinguished Engineer who has been working in the Austin, Texas lab since 1988, contributing to IBM's major software projects including OS/2, DB2®, AIX®, OpenDoc, Java, Component Broker, and WebSphere® Application Server. He is currently a member of the IBM Rational Software Chief Technology Officer team, influencing the strategic direction for products in the Rational brand and leading the solution definition for mobile application development. Leigh holds a BS degree in Computer Science from Nova University and a Masters degree in Computer Engineering from the University of Texas.

For more information

To learn more about IBM Rational solutions for mobile application development, please contact your IBM marketing representative or IBM Business Partner, or visit the following website: **ibm.com**/software/solutions/mobile-enterprise/

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

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
April 2012

[1] IGS paper, "Establishing an effective application strategy for your mobile enterprise," **ibm.com**/common/ssi/cgi-bin/ssialias?subtype=WH&infotype=SA&appname=GTSE_EN_OS_USEN_C&htmlfid=ENW03007USEN&attachment=ENW03007USEN.PDF

[2] Walker Royce, "Measuring Agility and Architectural Integrity" whitepaper, www.ijsi.org/ijsi/ch/reader/view_abstract.aspx?file_no=i92

Please Recycle

RAW14302-USEN-00