November, 2010

Software Engineering Issues for Mobile Application Development

Tony Wasserman, *Carnegie Mellon University*

Software Engineering Issues for Mobile Application Development

Anthony I. Wasserman
Carnegie Mellon Silicon Valley
Bldg. 23, M/S 23-14
Moffett Field, CA 94035 USA
+1 650 335 2807
tonyw@sv.cmu.edu

ABSTRACT
This paper provides an overview of important software engineering research issues related to the development of applications that run on mobile devices. Among the topics are development processes, tools, user interface design, application portability, quality, and security.

Categories and Subject Descriptors
D.2 [**Software Engineering**]: **D.2.2** Design Tools and Techniques

General Terms
Design, Reliability, Security, Human Factors

Keywords
Mobile devices, application development, software engineering, programming environments, user interface design, research agenda.

1. INTRODUCTION
While application development for mobile devices goes back at least 10 years, there has been exponential growth in mobile application development since the iPhone AppStore opened in July, 2008. Since then, device makers have created outlets for other mobile devices, including Android, BlackBerry, Nokia Ovi, Windows Phone, and more. Industry analysts estimate that there are more than 250,000 applications available through the various stores and marketplaces, some of which are available for multiple types of devices.

We have recently conducted a small survey of mobile developers [1], using available mobile developer forums to solicit respondents. A key goal of the survey was to gain a better understanding of development practices for mobile applications. Our conclusions included the following points:

1) most of the applications were relatively small, averaging several thousand lines of source code, with one or two developers responsible for conceiving, designing, and implementing the application;
2) there was a sharp divide between "native" applications, those that run entirely on the mobile device, and web applications, which have a small device-based client with execution occurring on a remote server;
3) developers adhered quite well to recommended sets of "best practices" but rarely used any formal development processes, and;
4) developers did very little organized tracking of their development efforts and gathered few metrics.

There are numerous comprehensive programming environments available for the major mobile platforms. Apple's iOS Dev Center offers the Xcode package, which includes an Interface Builder, an iPhone emulator, and a complete development environment that can be used across all Apple products [2]. For Android, developers can use the Android Development Tools plug-in [3] for the Eclipse programming environment [4]. For Windows Phone, developers can use a specialized version of Microsoft's Visual Studio environment [5]. Similarly, there are application development tools for BlackBerry, Symbian, and other platforms. In addition, there are now some cross-platform development tools, such as RhoMobile's Rhodes, MoSync, and PhoneGap, which can be used to create native applications on various brands of Smartphones. Along the same lines, Netbiscuits, Appcelerator, Kyte, and other companies provides tools and frameworks to support the creation of mobile web and hybrid sites using their SDK or one of the previously mentioned environments.

These powerful development tools and frameworks greatly simplify the task of implementing a mobile application. However, they are predominantly focused on the individual developer who is trying to create an application as quickly as possible. For small and medium-sized mobile applications that can be built (and easily updated) by a single developer, they represent a vast improvement on the previous generations of tools, and encourage developers to adhere to the important principles of abstraction and modularity that are built into the platform architectures.

However, as mobile applications become more complex, moving beyond inexpensive recreational applications to more business-critical uses, it will be essential to apply software engineering processes to assure the development of secure, high-quality mobile applications. While many "classic" software engineering techniques will transfer easily to the mobile application domain, there are other areas for new research and development. The remainder of this paper identifies some of these areas.

FoSER 2010, November 7-8, 2010, Santa Fe, New Mexico, USA.

2. SOFTWARE ENGINEERING AND MOBILE APPLICATION DEVELOPMENT

We define "software engineering" as a process by which an individual or team organizes and manages the creation of a software-intensive system, from concept through one or more formal releases.

2.1 What Makes Mobile Different?

In many respects, developing mobile applications is similar to software engineering for other embedded applications. Common issues include integration with device hardware, as well as traditional issues of security, performance, reliability, and storage limitations. However, mobile applications present some additional requirements that are less commonly found with traditional software applications, including:

1) Potential interaction with other applications – most embedded devices only have factory-installed software, but mobile devices may have numerous applications from varied sources, with the possibility of interactions among them;

2) Sensor handling – most modern mobile devices, e.g., "smartphones", include an accelerometer that responds to device movement, a touch screen that responds to numerous gestures, along with real and/or virtual keyboards, a global positioning system, a microphone usable by applications other than voice calls, one or more cameras, and multiple networking protocols;

3) Native and hybrid (mobile web) applications – most embedded devices use only software installed directly on the device, but mobile devices often include applications that invoke services over the telephone network or the Internet via a web browser and affect data and displays on the device;

4) Families of hardware and software platforms – most embedded devices execute code that is custom-built for the properties of that device, but mobile devices may have to support applications that were written for all of the varied devices supporting the operating system, and also for different versions of the operating system. An Android developer, for example, must decide whether to build a single application or multiple versions to run on the broad range of Android devices and operating system releases [6]

5) Security – most embedded devices are "closed", in the sense that there is no straightforward way to attack the embedded software and affect its operation, but mobile platforms are open, allowing the installation of new "malware" applications that can affect the overall operation of the device, including the surreptitious transmission of local data by such an application.

6) User interfaces – with a custom-built embedded application, the developer can control all aspects of the user experience, but a mobile application must share common elements of the user interface with other applications and must adhere to externally developed user interface guidelines, many of which are implemented in the software development kits (SDKs) that are part of the platform.

7) Complexity of testing – while native applications can be tested in a traditional manner or via a PC-based emulator, mobile web applications are particularly challenging to test. Not only do they have many of the same issues found in testing web applications, but they have the added issues associated with transmission through gateways and the telephone network

8) Power consumption – many aspects of an application affect its use of the device's power and thus the battery life of the device. Dedicated devices can be optimized for maximum battery life, but mobile applications may inadvertently make extensive use of battery-draining resources.

2.2 Best Practices

With all of the recent experience in creating mobile applications, much is known about how to build them and about how people use their devices and these applications.

At the same time, all but the largest and most complicated software and system development projects have moved away from a process-intensive approach toward a more agile approach, with the Scrum approach [7] and other agile techniques, e.g., test-driven development, finding widespread acceptance. That's particularly true of applications developed for the Web, where the development model relies on many successive releases of the evolving product. The Scrum development process is a sequence of short (2-4 weeks) "sprints" where a team addresses a set of tasks as a product increment, with each sprint addressing a "backlog" of requirements. Our survey of mobile developers [1] suggested that even individual developers are following a Scrum-like process as they develop mobile applications.

Above and beyond the process, though, is the systematic codification of knowledge about the best practices to follow for application development. The World Wide Web Consortium has issued a candidate set of recommendations for mobile web (not native) applications [8]. Apple has published an iPhone Application Programming Guide [9] with guidelines for various aspects of iPhone development. The Developer's Guide for Android includes a Best Practices section that addresses application compatibility, user interface guidelines, and designing for performance and responsiveness, among other things [10].

In short, developers can find a lot of guidance to assist them with programming their applications. Platform developers have drawn on decades of software engineering knowledge to create software architectures and SDKs that provide developers with access to needed device resources. However, these technical aspects don't address the larger issues of creating large-scale applications.

2.3 Finding the Balance

One of the long-term challenges in every engineering discipline is "scaling up": finding appropriate techniques for managing increasingly complex projects. Approaches that work well for an individual engineer don't always work when the tasks of a project are divided among members of a team. The team (and any supervisory management) need mechanisms for coordination and reporting. The added complexity of larger projects often demands greater attention to [changing] requirements, product architectures, and testing, as well as to key project properties, such as robustness, usability, reliability, and more.

For mobile devices and their applications, the software engineering process must not only be aware of the hardware device properties, but must also address project management issues and the unique aspects of mobile application development noted above.

Many large-scale and enterprise-oriented mobile applications will be part of a product family. These applications will often be mobile web applications, rather than native mobile applications, and will often complement or augment an existing application. As a result, development of the mobile application will typically be done within the context of the overall software development effort, thus providing a management framework for the mobile application. However, the unique qualities of the mobile environment makes it important not to treat the mobile application as an afterthought, but rather as an independent task with its own software engineering process and product requirements.

3. A RESEARCH AGENDA FOR MOBILE SOFTWARE ENGINEERING

Despite the development of 300,000+ mobile applications, there's still not much formal research around their engineering processes. The existing body of knowledge is highly pragmatic, with lots of guidelines and many pieces of sample code as examples. In this section, we identify some of the most promising areas for software engineering research related to development of mobile applications

3.1 The User Experience

Using a mobile device is different from working with a desktop or laptop computer. While gestures, sensors, and location data may be used in game consoles and traditional computers, they play a dominant role in many mobile applications. The smaller display and different styles of user interaction also have a major impact on interaction design for mobile applications, which in turn has a strong influence on application development. The mobile user interface paradigm is based around widgets, touch, physical motion, and keyboards (physical and virtual) rather than the familiar WIMP (Windows, Icons, Menus, Pointer) interface style of Apple's iOS and Microsoft Windows. Other context dependencies may also play a role in the user experience, including such aspects as physical location, proximity to other mobile devices, and the activation of various device features

Mobile platforms include their own UI libraries and guidelines, so native applications for a device will share a common "look and feel." It's in the interest of the application developer to adhere to platform standards, especially on touch-screen devices, where users expect to use the platform's standard set of gestures, which differs for each platform.

With the challenge of making the best possible use of limited screen space, user interface design takes on greater importance than ever. Mobile users are often seeking to quickly complete a simple task, and can't take advantage of the full range of functionality provided by a traditional Web application.

The user interfaces for mobile web applications may borrow from traditional web applications, but must often be redesigned to highlight the most commonly used functions and to make most effective use of the screen and the mobile user interface paradigm, including both the user input and the associated motion and location information.

These observations raise some research issues, including:

1) How does one determine which functions should be present in a mobile version of a traditional application? Are there techniques that can assure the maximum reuse of code among different versions?
2) What is the comparable effort to build a native mobile application (or a set of them for different platforms) compared to a mobile web application? Is there a measurable difference in user satisfaction or productivity with either of these?
3) Is there a need for specialized scenario development processes and tools for mobile applications? Does the mobile UI require a different contextual design process to support a different set of use cases?
4) How does a software designer integrate the various forms of input and sensor data in application design?

The user experience is also strongly affected by other industrial design issues related to the device itself, e.g., weight and size, but these items are largely outside the domain of software development, and not discussed further here

3.2 Non-functional Requirements

The success of any application, mobile or otherwise, depends on a lengthy list of non-functional qualities. Among those most relevant to mobile applications are performance (efficient use of device resources, responsiveness, scalability), reliability (robustness, connectivity, stability),quality (usability, installability), and security. Many of these issues have been addressed for web applications, and that knowledge provides an excellent starting point for studying mobile application requirements.

The mobile environment, with its dependence on different kinds of networks, differs from traditional environments and thus raises some new research questions, such as:

1) Do mobile web applications behave differently when connected using the telephone network (3G, 4G) than when using an 802.11 (WiFi) or 802.16 (WiMax) connection? Are there differences in security? Is there a significant difference in responsiveness? Are traditional fallback and exception-handling techniques adequate, or does the higher likelihood of a dropped connection (or intermittent connectivity) require additional mechanisms?
2) Are there new techniques needed for assuring data integrity, or will the synchronization techniques from traditional client-server computing suffice? Does potential loss of connectivity or battery power represent a risk to program and/or data integrity if such an event occurs during a transaction or system update?
3) Should applications be designed differently depending on the speed of the network on which they are being used? In Asia, some countries offer rates of 50Mb or higher, while typical speeds in the US, even with 3G networks, are below 1 Mb.
4) How does a developer create applications that will maximize battery life and resource usage?

Again, these questions are just a small subset of a broad range of research questions that need further study.

3.3 Processes, Tools, and Architecture

As mobile applications become more complex and mission-critical, development organizations must introduce processes that address more aspects of the development process than are covered in today's agile processes and development environments. As previously noted, the user experience is especially critical, so there is a greater need to create prototypes of the user interface(s), particularly when multiple devices will be supported.

Testing is another important area for mobile software engineering research. One question involves the development of testing methods for product families, such as Android devices. It's insufficient to merely test an Android application on an emulator; it must be tested across many different Android devices running different versions of the operating system on various telecom networks, perhaps with l10n and i18n options. Integrated test suites would simplify this process.

Another area for research involves application maintenance in the rapidly changing world of mobile platforms. While "early-adopter" consumers are often willing to update their device and their applications, most enterprise users are less likely to do so. In many cases, their companies will have policies discouraging them from doing so, as can be seen by the slow enterprise transition away from Windows XP and Office 2003. One particularly interesting question involves the use of virtualization technology on these devices as a way to support various platforms.

Finally, application development and deployment is moving toward the "cloud". This new computing paradigm will not only affect development processes and tools, but also application architectures.

3.4 Portability

Application developers quickly developed apps for the iPhone platform following Apple's creation of the AppStore. As noted above, other providers of mobile platforms and devices have done the same (or are in the process of doing so). An important issue for the application developer is to decide which platform(s) to support in the highly fragmented world of mobile development. Today, there are at least five important platforms (iPhone, Android, BlackBerry, Windows Phone, Symbian).

From the standpoint of the application developer, it's quite expensive to support multiple platforms, especially when there are multiple versions and variants for each of them. The application developer has several options:

1) develop for a single platform only and use, to the extent possible, a common subset of the features available across all variants and versions of that platform; thus, for example, the developer would have only a single code base for an application that would run on different versions of the iPhone, the iPad, and possibly the iPod Touch. While that approach would simplify the developer's work, the resulting application would not be able to take advantage of all of the differentiating features of each device ;

2) develop native applications for each platform and variant, trading off the development and maintenance costs against the ability to optimize the application for each platform.

3) develop mobile web applications, thus minimizing the amount of native code for each platform; it remains uncertain whether this approach will meet the needs of the market, or;

4) use one or more layer(s) of abstraction that can map a "write once" application into native executable programs that will run on multiple platforms.

Each of these approaches presents a set of research questions, and suggests the need for customized tools to support cross-platform development and testing.

4. CONCLUSION

The items discussed in Section 3 are only a subset of the possible research topics in software engineering for mobile applications, but serve to indicate the breadth of research needs and opportunities in this emerging field.

While the large number of mobile applications makes it appear that software development processes for them are well understood, there remain a large number of complex issues where further work is needed. In addition, there is a mobile "angle" to almost every aspect of software engineering research, where the characteristics of mobile applications and their operating environments present a new or different set of research issues

5. REFERENCES

[1] Agrawal, S. and A.I. Wasserman, "Mobile Application Development: A Developer Survey", submitted for publication, 2010

[2] Apple Developer Connection. http://developer.apple.com/iphone/index.action. Accessed on 6 September 2010.

[3] Android Developer site. http://developer.android.com. Accessed on 6 September 2010

[4] Eclipse web site. http://eclipse.org. Accessed on 6 September 2010.

[5] Windows Phone developer site http://developer.windowsphone.com/windows-phone-7-series/ Accessed 6 September 2010.

[6] Fring, Brian. 2009. *Mobile Design and Development*. O'Reilly.

[7] Schwaber, K. 2004. *Agile Project Management with Scrum*. Microsoft Press.

[8] World Wide Web Consortium, Mobile Web Application Best Practices W3C Working Draft, 13 July 2010. http://www.w3.org/TR/mwabp/ Accessed on 6 September 2010.

[9] Apple. iPhone Application Programming Guide. http://developer.apple.com/iphone/library/navigation/index.html. Accessed on 6 September 2010.

[10] Android Developers. The Developer's Guide. http://developer.android.com/guide/index.html Accessed on 31 May 2010.