2nd Annual Workshop on Software Engineering for Mobile Application Development

www.mobileSEworkshop.org

MobiCASE '11 - Santa Monica, CA, USA
27 October 2011

2nd Annual Workshop on
Software Engineering for Mobile Application Development

Santa Monica, CA – 27 October 2011

Overview Paper

Workshop Position Papers

Workshop Organization

Anthony I. Wasserman, Carnegie Mellon Silicon Valley (Chair)
Ray Bareiss, Carnegie Mellon Silicon Valley
Ralf Carbon, Fraunhofer IESE

Sponsor: Appcelerator, Inc.

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.

Improving Mobile Application Development

Ray Bareiss and Todd Sedano
Carnegie Mellon University
Silicon Valley
{ray.bareiss, todd.sedano}@sv.cmu.edu

Many mobile app developers act more like end-user programmers – professionals in disciplines other than software development who create computer programs as pragmatic tools to solve work-related problems – than like professional software engineers. They want the quick gratification of creating a program rather than what they view as the drudgery of following engineering processes. They focus on the construction phase of software development, slighting other phases such as requirements, design, testing, and debugging.

Ko, et al, have drawn a clear contrast [1]:

Software Engineering Activity	Professional SE	End-user SE
Requirements	*explicit*	*implicit*
Specifications	*explicit*	*implicit*
Reuse	*planned*	*unplanned*
Testing and Verification	*cautious*	*overconfident*
Debugging	*systematic*	*opportunistic*

The end-user software engineering research community is actively working to improve the quality of end-user-produced software [1,2,3]. End-user software engineering is defined as systematic and disciplined activities performed by end-user programmers that address software quality concerns [1]. We can draw upon the lessons of end-user software engineering research to devise a lightweight set of tools, techniques, and processes that mobile app developers may recognize as being worth additional overhead because of improvements in software quality, ideally leading to higher app store ratings and greater customer adoption. (Most mobile app developers are keenly aware of their ratings, and any techniques that demonstrably increase ratings will be easy to "sell" to the community.)

Research in end-user software engineering has primarily addressed requirements, design, reuse, verification, and testing.

In reality, the requirements and design processes are tightly intertwined and are, perhaps best treated as one. This combined requirements and design process maybe the easiest one to sell to app developers: In the abstract, design is sexy and better designed apps that better meet the user's needs are likely to have higher adoption rates and, thus, revenues. A lightweight, user-centered requirements and design process might comprise Rapid Contextual Inquiry [4], persona- and scenario-based design [5], adherence to platform guidelines [6,7], early prototyping, and

lightweight user testing [8]. (Note that user testing is an important way to discover requirements in addition to providing usability feedback, per se.)

Reuse can comprise both reuse of patterns and of code. Pattern reuse is somewhat encouraged at the user interface level by platform design guidelines which are, at least, loosely enforced by the two major app stores [6,7]; in particular, Apple tightly controls their API to encourage developers to use the phone in a certain way. Code reuse for the construction of human and, especially, sensor interfaces is a relatively easy sell given the complexity of coding such interfaces. The complexity of cloud back-ends, which are increasingly important to nontrivial apps, may also encourage reuse of both architectures and code. The key to promoting reuse may simply be making developers more aware of available patterns and code.

Promotion of a rigorous testing process is more problematic because testing is arguably the least glamorous aspect of software development. Much research on end-user software engineering has focused on this area. The most promising approach seems to be the incorporation of visual indicators into development environments to encourage software testing, e.g., WYSIWYT (what you see is what you test) [9]; ironically, common development platforms do not make automated testing easy. Research also suggests the promise of a Surprise-Explain-Reward approach [10] to rouse a developer's curiosity and then concern, thus promoting engagement in a systematic testing process such as statement and branch coverage testing.

Perhaps the most problematic goal is to convince app developers to engage in some degree of formal verification of their code. Despite its long-time popularity in software engineering curricula, the use of verification techniques by professional software engineers remains practically nonexistent, so the prospect of getting app developers to adopt verification seems remote indeed.

Over the long-term, there is another approach to improving software quality. The software craftsmanship movement appeals to developers' desire to be recognized as great programmers by their peers [11]. Developers are encouraged to practice areas in which they are weak in order to become experts in their field [12]. Some programmers convert their practices into a codified kata so that other developers can more easily learn the same lessons. While katas are prevalent for most programming languages, there is a dearth of specific mobile katas. (At the time of writing, we could only find one.) Developers also pair program with other developers for the express purpose of improving their craft, and in fact, some will even travel around the world on a "journeyman tour" [13].

This paper has proposed a set of techniques, tools, and processes, drawn in large part from end-user software engineering research, that the authors believe to compose a reasonable approach to improving the quality of mobile apps. The remaining questions are "Are developers willing to learn these techniques?" and "Are they willing to use them consistently?" One way to explore this question is to

offer a course in mobile application development that embodies these techniques (assuming that appropriate tools are in place as noted earlier in the paper). A project-based educational approach using the pedagogy pioneered by Carnegie Mellon University - Silicon Valley seems most appropriate for the targeted students [14]. Such a course could be offered to matriculating and nonmatriculating students at our campus. However, it might reach a wider audience at a community college. Finally, the best approach of all might be to use principled mobile app development to interest young people in careers in software development via a "summer camp" format. We hope to return in the near future to report on the development of such a course.

References

[1] Ko, A. J., Abraham R., Beckwith L., Blackwell A., Burnett M.M., Erwig M., Scaffidi C., Lawrence J., Lieberman H., Myers B.A., Rosson M.B., Rothermel G., Shaw M. and Wiedenbeck S., "The State of the Art in End-User Software Engineering," ACM Computing Surveys, to appear.

[2] Burnett, M., 'What is end-user software engineering and why does it matter?" In End User Development. Proceedings of the 2nd International Symposium, IS-EUD 2009, Wulf, V. and Burnett, M. (Eds.) Siegen, Germany, March 2-4, 2009. LNCS Volume 5435/2009. Dordrecht: Springer, pp. 15-28, 2009.

[3] Dwivedi, V., "End-User Software Engineering" (PowerPoint presentation) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture27-End%20User%20Software%20Engineering.pptx

[4] Holtzblatt, K., Burns Wendell, J., and Wood, S., Rapid Contextual Design, San Francisco: Morgan Kaufmann, 2005.

[5] Goodwin, K. and Cooper, A., Designing for the Digital Age: How to Create Human-Centered Products and Services, San Francisco: Wiley, 2009.

[6] Anonymous, Apple iOS Human Interface Guidelines, http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html

[7] Anonymous, Android User Interface Guidelines, http://developer.android.com/guide/practices/ui_guidelines/index.html

[8] Krug, S., Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems, Berkeley: New Riders, 2010. (Also see http://sensible.com, especially the video of an example usability test.)

[9] Rothermel G., Burnett M., Li L., Dupis C. and Shertov A., "A Methodology for testing spreadsheets.," ACM Transactions on Software Engineering Methodologies, 10(1), 110-147, 2001.

[10] Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., Durham, M., Rothermel, G.: Harnessing Curiosity to Increase Correctness in End-User Programming. In: ACM Conference on Human Factors in Computing Systems. ACM, New York, 2003.

[11] Anonymous, Manifesto for Software Craftsmanship, http://manifesto.softwarecraftsmanship.org

[12] Ericsson, K., Prietula, M., Cokely, E.: "The Making of an Expert." Harvard Business Review. 2007.

[13] Hoover, Dave. "Dave Hoover's Journeyman Tour" http://nuts.redsquirrel.com/post/1181144648/dave-hoovers-journeyman-tour

[14] Bareiss, R. and Sedano, T. , Developing Software Engineering Leaders, Proceedings of the First International Symposium on Tangible Software Engineering Education (STANS-09 in Tokyo, Japan), October 2009.

Preparing Mobile Software Development Processes to Meet Mission-Critical Requirements

Position Paper

Luis Corral, Alberto Sillitti, Giancarlo Succi
Center for Applied Software Engineering, Free University of Bozen-Bolzano
Piazza Domenicani 3, I-39100 Bozen-Bolzano, Italy
Luis.Corral@stud-inf.unibz.it, {Alberto.Sillitti, Giancarlo.Succi} @unibz.it

1 Introduction

Mobile systems have evolved to a level where they are required to carry out their operations in a wide range of environments, from voice communication, networking and entertainment to complex business applications. Software running on mobile devices has as well grown to a point in which it has earned a major role and impact on the overall performance and capability of a system. This situation attracts a genuine interest of knowing in a more detailed fashion the conditions under which mobile software systems are conceived, designed, implemented, and maintained, both from process and product point of views.

In parallel, "mission-critical" is a recurrent concept in Software Engineering that refers to software systems in which a malfunction may lead to injuries, loss of lives or loss of significant investment. As consequence, these systems require the highest care through all steps of the software development cycle [1]. Application areas of mission-critical software comprehend manufacturing control, transportation systems, weapon delivery systems, space programs, air traffic control, medical systems and many others [2].

The continuous growth in capabilities in software applications running in mobile devices allows them to become an attractive platform to be part of more complex systems, including safety-critical or mission-critical systems. As mentioned above, due to the nature of these environments, such mobile software system would be required to meet strict assurance practices with regard to safety, security and performance, being subject to attain certifications from competent authorities before entering into service.

Currently we count on several software processes and quality management systems intended to support the development of general-purpose (e.g. CMMi, ISO 12207, ISO 15504) or application-specific (e.g. RTCA/DO-178B, ISO 62304, EN 50128) software. These models were conceived and developed without considering restrictions given by a specific target platform, and are general enough to accommodate any execution environments (e.g. embedded, desktop, etc.), so they are prepared to suit the needs and restrictions of mobile equipment.

On the other hand, the number of published development models specific for mobile software systems (e.g. Mobile-D [3] or RaPiD7 [4]) is still limited, and they present addressable shortcomings. These models converge in the selection of Agile methodologies to provide a development framework, calling attention on the necessity of adapting the processes and practices to the evolving needs of mobile software, shorting the development cycles. As a common characteristic, they have claimed little level of criticality on mobile applications, so Agile quality assurance practices do not necessarily fulfill the requirements imposed by application-critical standards or certification procedures [5].

In summary, currently it is not available a methodology or framework to manage the assurance of mobile software so as to produce application-critical, certifiable software, taking into account the needs, boundaries, and particular conditions given by a mobile environment. Such conditions may impact the quality of the software product within a mission-critical system, and deserve to be carefully considered in addition to the practices already dictated by safety-critical software development standards.

We believe that it is necessary to study (i.e. survey, leverage) current software assurance practices adopted by mobile development, and analyze the existing gap between them and mission-critical software standards. After knowing such distance, we may identify the ways in which current practices can be strengthened or adapted to have a robust mobile software assurance process that positively aids mobile developers to produce software able to be incorporated into a certifiable mission-critical environment.

2 State of the Art

It is still difficult to find a disciplined, strong and dependable development process for mobile applications [6]. Abrahamsson et al. [3] discussed that the development of mobile applications is a challenging task due to the specific demands and technical constraints found on a mobile environment. By then, very little was known about the suitability of the different development processes for mobile application development. More recently, in the work of Franke and Weise [7] it is set the question "Does software quality assessment, in particular testing, differ between desktop and mobile software?" after which they evaluate common assurance practices to be applied to mobile software just in the way they are used for other environments.

A first attempt to design a software development methodology for mobile devices is introduced by Abrahamsson et al. [3] in the methodology called Mobile-D (inspired by Extreme Programming). Mobile-D is proposed as an Agile approach to mobile application development, later endorsed by independent works by Abrahamsson [8] and by Rahimian and Ramsin [9]. In the same sense, RaPiD7 (Rapid production of documentation in 7 steps) [4] was developed and applied at Nokia; it is as well inspired by Agile practices and focuses on describing human interaction, planning and documenting in software projects, both related and unrelated to mobile development. In all cases, the incorporation of Agile practices into mobile software processes claimed little criticality level on mobile applications, and a strives for a continuous reduction of the development cycles due to the constant evolution and change of requirements, platforms and users.

These works represent, indeed, an interesting approach to be followed; nonetheless, due to their Agile nature they may fail on establishing a clear software assurance strategy based on traceable plans, procedures, records and reports imposed by application-critical standards and certification authorities, typically based on plan-driven practices.

3 Open Items on Mobile Software Assurance to Meet Mission-Critical Requirements

A significant amount of work is yet to be done to combine mobile software development processes with application-critical standards. The goal is to take advantage of our experience on Agile and mission-critical development to achieve the configuration of a robust framework that learns from both traditional and innovative software engineering practices that have proven their value on the development of conventional and mission-critical software:

- Current mobile software methodologies based on Agile paradigms have to be adapted to meet the needs of application-critical standards (i.e. adding traceability records, documentation, prescribing independence between roles, etc.).
- A number of practices from Agile methodologies (e.g. refactoring, pair programming, customer involvement) should be incorporated to the development of critical software thanks to the added value and positive impact they give to the product's quality.

The range of applications deployed on cellular telephones and tablets is continuously growing, moving this family of devices towards becoming a primary end-user computing equipment. We are not far from having mobile telephones as software platforms fully operating an aerospace system (e.g. maintenance crews operating handheld gadgets), healthcare systems (e.g. monitors on cellphones with direct communication to hospitals), or similar applications that require to meet safety-critical standards and obtain government certifications previous to enter into service.

4 Vision and Research Approach

Our vision is to innovate, extend and strengthen current mobile software frameworks to assure the quality of the product to the most stringent safety-critical standards and certification requirements. As software for mobile devices grows in capacity and dependability, the lack of a software assurance process for mobile applications is more noticeable, and the unveiling of a proper one becomes an imperative.

To achieve this goal, we have identified as a starting point the adapting of Agile methodologies, based on the fact that such methodologies are now the most accepted approach for managing mobile software development processes. Analyzing independently current mobile software development methodologies and Agile practices, we have identified that they lack of several practices and conditions required by certification processes on application-critical software, for example, rigorous project documentation (plans and results), traceability records, independence between major roles (e.g., independence between development and testing individuals) and other requirements.

Our research approach is to leverage the requirements set by the most important development standards for application-critical software (e.g., RTCA/DO-178B for airborne applications, ISO 62304 for medical systems, and EN 50128 for rail systems) and conduct a gap analysis that will allow us to identify the major limitations shown by current mobile development methodologies with respect to safety-critical standards. In addition, we will incorporate further failure modes given by the mobile environment itself (e.g., failures in communication networks, limited computational resources, and others) to complement the safety considerations to be born in mind when a mobile device and its software are full participants on an application-critical system.

After this analysis, we will count on a series of recommendations (namely practices, records and requirements) to be added to the existing mobile software development methodologies. These recommendations are not intended to replace current development practices for mission-critical software, but they will represent a reliable guideline to strengthen different practices throughout the mobile software development, with the objective that processes, products and documentation are suitable to meet safety-critical certification requirements, and most important, assure the development of a software product that will be deployed in an environment in which there is no room to afford any failure.

References

[1] McQuaid, P. 2009. *Software Disasters: What have we learned?*. Software Quality Professional, volume 11, issue 3. ISSN 1522-0542. Pp. 28. The American Society for Quality.

[2] Medikonda, B.S., Seetah Ramaiah, P. 2009. *An approach to modeling software safety in safety-critical systems*. Journal of Computer Science 5, 2009. Pp. 311-332. ISSN 1549-3636. © Science Publications.

[3] Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O. 2004. *Mobile-D: An Agile Approach for Mobile Application Development.* OOPSLA'04.

[4] Kylmäkoski, R. 2005. *RaPiD7: A Collaborative Method for the Planning Activities in Software Engineering - Industrial Experiment*. ISBN 952-15-1517-1. Nokia.

[5] Ambler, S. 2005. *Quality in an Agile World* in Software Quality Professional Journal. volume. 7, issue. 4. ISSN 1522-0542. Pp. 34-40. The American Society for Quality.

[6] Glissmann, S., Smolnik, S., Schierholz, R., Kolbe, L., Brenner, W. 2005. *Proposition of an M-Business Procedure Model for the Development of Mobile User Interfaces*. IEEE.

[7] Franke, D. and Weise, C. 2011. *Providing a Software Quality Framework for Testing of Mobile Applications.* Fourth IEEE International Conference on Software Testing, Verification and Validation.

[8] Abrahamsson, P. 2005. *Mobile software development – the business opportunity of today.* Proceedings of the International Conference on Software Development, 2005. Reykjavik, Iceland, pp. 20-23.

[9] Rahimian, V. Ramsin, R. 2008. *Designing an Agile Methodology for Mobile Software Development: A Hybrid Method Engineering Approach.* Second International Conference on Research Challenges in Information Science.

Mobile Business Applications must be thoroughly engineered!

Ralf Carbon
Head of the Research Area “Business Goes Mobile”
Fraunhofer IESE
Kaiserslautern, Germany
ralf.carbon@iese.fraunhofer.de

Steffen Hess
Fraunhofer IESE
Kaiserslautern, Germany
steffen.hess@iese.fraunhofer.de

Ralf Carbon works in applied Software Engineering research since 2002 after he received his diploma in computer science from the University of Kaiserslautern, Germany. He performed many projects in the area of software architecture, especially in the domain of large, workflow-based information systems. Since 2009, he performs projects on mobile business applications covering their complete life-cycle. Since 2010, he is the head of the newly established research area “Business Goes Mobile” at Fraunhofer IESE. The research area currently consists of a team of 10 researchers working on mobile software engineering projects.

Steffen Hess works at Fraunhofer IESE in the field of usability and user experience since 2004. He worked first as a student employee and after receiving his diploma in industrial engineering as researcher and practitioner. Already in his diploma thesis he worked in the field of mobile software engineering – comparing different mobile open source and proprietary operation system ecosystems. Since 2009, he performed many projects on mobile business applications for different customers in different areas. Performed work in this context covers especially conception and prototyping of apps using a lightweight engineering approach.

1. How does traditional software engineering relate to the engineering of mobile applications and systems?

In the research area “Business Goes Mobile” at Fraunhofer IESE, we develop methods to systematically engineer mobile business applications (apps) and transfer them to industry. Mobile business apps are provided by organizations to offer valuable mobile services to their customers (B2C scenario) or equip their own mobile workers with appropriate workflow support (B2B scenario). We supported, for instance, customers in the agricultural domain to develop mobile business apps for farmers, contractors, and field workers and in the airline domain to build mobile business apps for passengers, pilots, and service personnel. The examples show, that mobile business apps are used in business-critical situations and therefore need to provide the required functionality with guaranteed quality. Especially a great user experience, security, but also flexibility with respect to future extensions are beyond the key quality requirements.

In our industrial projects we experienced many insufficient apps. The main reason for mobile business apps of poor functionality and quality is that such organizations do not thoroughly engineer them. Driven by tough time to market constraints, they suppose mobile business apps can be implemented quickly without precisely specified requirements and without thorough UI, interaction, and architectural design.

Our approach is reuse parts of "traditional" Fraunhofer IESE software engineering approaches we applied in many domains for many years, but to tailor them where required and to complement them by, for instance, solution patterns and lessons learned from the mobile domain.

We decided on certain guiding principles for mobile software engineering that support us in tailoring our existing methods to address key challenges of mobile business apps:

Be user-centric: Heavily involve the end-user throughout the engineering process of mobile business apps.
Be lightweight: Run through all major activities of a typical software engineering approach, but focus on the production of key artifacts.
Be iterative: Perform short iterations to get quick and continuous feedback (especially from end-users) and continuously improve your mobile business apps based on that feedback.
Be integrative: Take care of close interaction between major stakeholders in the development team, for instance, requirements engineers, UI and interaction designers, and architects.

While these principles are generally valuable for all kind of Software Engineering projects, we want to specifically point out the importance of them in the case of Mobile Software Engineering projects.

2. What are the distinguishing features of mobile software specification, architecture, development and testing that need special attention, skills, or innovation?

Currently, our research in mobile software engineering mainly focuses on requirements engineering and architectural design.

Requirements engineering for mobile business apps must be performed quickly, must focus on the main stakeholders, especially the end-users and their main requirements, and results must be documented precisely but in a lightweight fashion. We try to accomplish this, for instance, by performing a one day requirements engineering workshop with the major stakeholders per release (especially involving the end-users), with a fixed agenda, and a documentation template that is reduced to the essential requirements artifacts.

Architectural design for mobile business apps should be focused based on typical quality requirements of the mobile domain. If such typical domain requirements and potential solutions can be represented adequately in a reusable form, the quality of the architecture and the efficiency of the architectural design process can be improved to better fulfill the tough time to market constraints of mobile business apps and guarantee a certain quality of the product. We try to capture quality requirements and architectural solutions in a platform independent way and if required describe iOS or Android specific instances.

3. What should be the suggested focus and agenda for mobile software engineering research and education? What new knowledge and skills do practitioners need most?

Mobile software engineering research and education should focus on developing and teaching methods considering the specific challenges of mobile application development. Such challenges are, for instance, short time to market and high quality, especially user experience, security, and flexibility. According to our philosophy mentioned above, requirements engineering, UI and interaction design, or architectural design approaches for mobile business apps should be user-centric, lightweight, iterative, and integrative.

Mobile software engineering methods need to provide answers to practitioner, for instance, with respect to the following questions:

- How can we achieve a great user experience?
- How can we design for multiple platforms?
- What is the right scope of an app?
- What do we need to consider when integrating an app in an existing IT infrastructure?
- …

Mobile-Enabling Enterprise Business Applications using Model-Driven Engineering Techniques

Suman Roychoudhury and Vinay Kulkarni
{suman.roychoudhury, vinay.vkulkarni}@tcs.com
Tata Research Development and Design Center,
Tata Consultancy Services,
Pune – 411013, India

Abstract

According to a recent Gartner report it is predicted that the number of the combined installed base of smartphones will exceed the installed base of PCs by 2013. This indicates that there would be a huge market of mobile-enablement of existing enterprise systems. Current state-of-the-art within the industry is to develop a fresh channel application for all mobile platforms of interest. Instead we propose to use model-based generative approach to deliver the same specification onto multiple mobile platforms, i.e., an ability to model the service being invoked, the interfacing screen, the data along with the interaction protocol and the capability to generate code from its specification. We have initial proof-of-concept results that verify the validity of our proposed technique.

1 Introduction

The advent of rich mobile computing platforms [1] along with continuous improvement in the underlying hardware capabilities and network infrastructure [2] have resulted in a steady demand for mobile computing applications. Although early adoptions were focused towards entertainment centric applications (e.g., gaming, music, movies, videos), recent trends suggest a deeper interest in mobile enablement of enterprise business applications (EBA). For example, financial applications such as mobile banking, portfolio management, mobile brokerage etc are slowly gaining acceptance among mobile users [3]. Financial institutions and business houses who were earlier cautious on providing customer specific services on mobile devices are now following a more liberal path. Moreover, a recent Gartner report [4] predicts a steep incline in the number of smartphone users over PC users and this indicates a paradigm shift from the traditional desktop-based computing to mobile computing. However, this sudden rise in demand for mobile enablement of business applications may come at a high price. This is because of the diversity in hardware, platform, environment, user interaction etc., such applications have to be rewritten for every specific target context although their core functionally remains exactly the same. This increases the development cost of creating "multiple same" application thereby imposing a heavy financial burden on business institutions and also slows down the application's time-to-market. Therefore in this position paper, we explore novel software engineering techniques that can transform EBAs to be mobile compliant.

2 Enterprise Business Applications

Typically, any n-tier enterprise business application can be modeled using 3 layers, namely, the business logic, the data and the user interface (UI) layer. The business logic is generally encapsulated within a set of services that interacts with the data layer. The UI layer typically constitutes the front-end of the system that involves interaction with the user interface. Fig 1a depicts the above scenario that shows how the interaction among these layers takes place via simple request and response protocol.

Fig 1b presents the same EBA model from a mobility perspective. Some of the concerns that are associated in modeling the data part of any EBA are synchronization, consistency, persistent storage etc. On similar lines, the concerns that are associated with modeling the UI part are the interaction objects (i.e., widgets), their mode of operation (modality) and their interaction pattern. For example, there may be multiple ways a user can browse a product catalog, but it is also important to find the most effective or optimal way the user can browse through the same information depending on the device and the platform (mobile) he/she chooses to use. Therefore interaction pattern plays an important role in the design of UI part of any application. Finally, the services that are generally distributed in nature are invoked via different communication protocols, for example SOAP, SMS, IVR etc.

Fig 1a: 3-Tier Enterprise Application Model

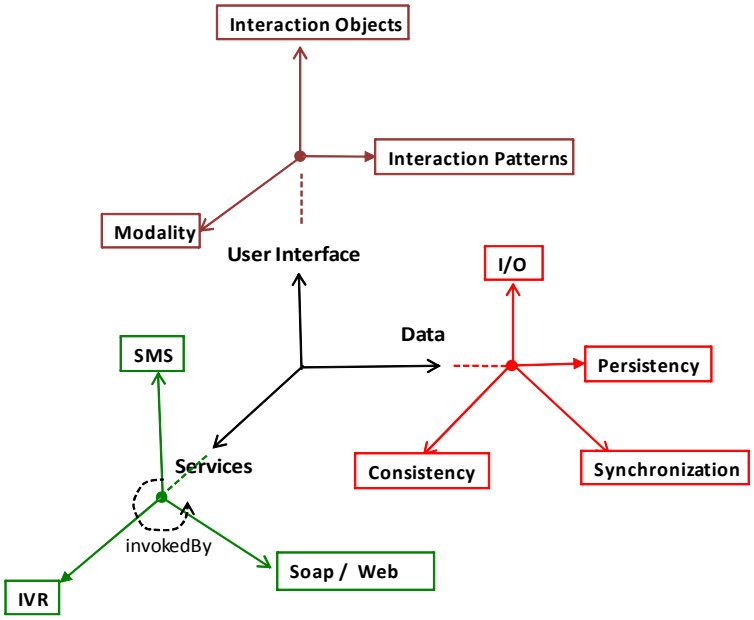

Fig 1b: Multi-dimensional view of a Business Application

3 Mobile-Enabling EBAs

For any enterprise that offers a set of services on the desktop platform, the very first question that comes into mind is - *How do I transform my existing enterprise application to be mobile enabled?* Following this question, the set of related questions are:

- How can the transformed application be deployed on any mobile platform or device?
- How to provide the best user experience on the target platform or handset?
- How to deliver mobile applications that are simpler to build, quicker time-to-market and relatively easy to maintain?

To answer the above questions, we reflect into our past software engineering practice and experience. For the past several years, we are using model-driven engineering (MDE) techniques to address the need of developing enterprise application from high-level models [5]. Models capture the design and specification of any application and raise the level of abstraction from code-centric way of application development to model-centric way of application development. Fig 2 presents our current MDE technique for constructing n-tier enterprise business applications (EBA). It shows three primary constituents (i.e., Data, UI and Services) of an EBA grouped across three principal technical spaces – namely the specification space, the generation space and the implementation space. Out of these three technical spaces, the specification space is used to capture the high-level specification of an application whereas the generation space is composed of model-based code generators that take the model specifications as input and produces the low-level implementation as output. Finally the target platform compiler(s) compiles all the generated artifacts which are then packaged, deployed and run on the runtime system.

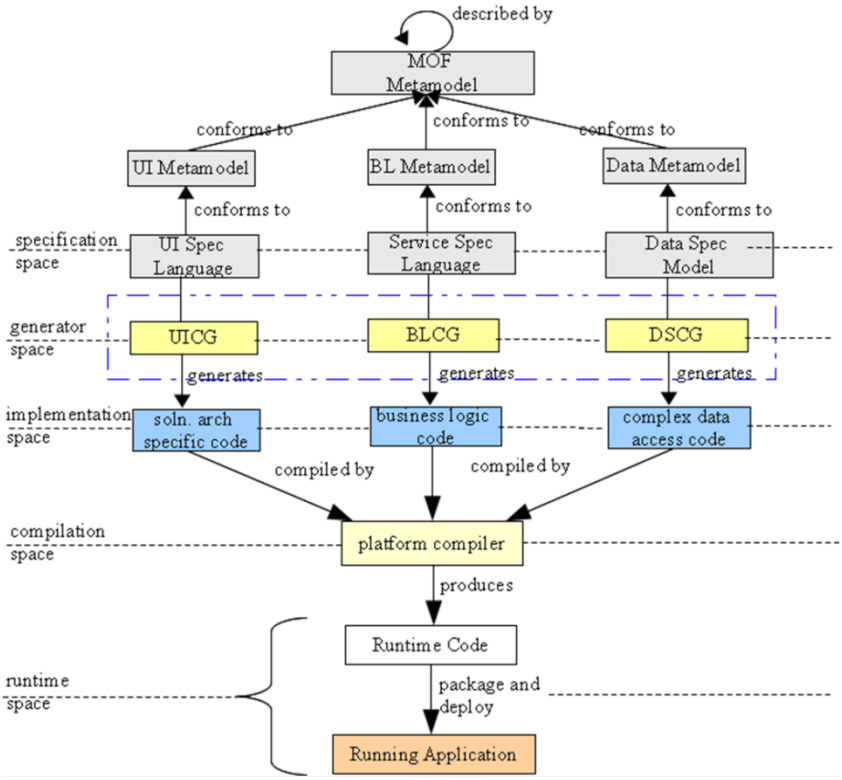

Fig 2: Building Enterprise Applications using MDE Techniques

We believe the principles, the techniques and the methodologies that we used in building EBAs can well be applied to mobile-enable business applications. From our initial experience in manually transforming EBAs to be mobile-enabled, we have observed that for the same application, the functionality across the data and the services layer generally do not change from one mobile platform to another. However, due to the variability in the UI implementation space across various mobile platforms, one needs to redevelop the entire UI part of the application. Especially, in case of native UIs (e.g., Java in Android, ObjectiveC in iOS), that generally offers a richer user experience than web-based UIs (i.e., using webkit and embedded browser), the problem is several times compounded. For example, Android supports several widgets that are not supported in iOS (e.g., [1]*checkbox*). Like this, there are several other variations in the native UI layer across mobile platforms. These delta variations in the UI specification for the same application are depicted in Fig 3 as *UI Spec α* and *UI Spec β*. Thus, in order to capture such Δ variations in the UI layer, the individual platform specific code generators are configured via a parametrization process that takes into consideration the abstract UI mapping models across multiple mobile platforms. The bottom of Fig 3 shows the corresponding platform specific applications that were generated from the same application specification.

4 Conclusion

There are several challenges in building mobile applications catering to different functional and non-functional requirements of a business application (e.g., sensor management, energy management etc). However, in this position paper, we tried to address a particular challenge in mobile-enabling existing EBAs by reusing most of existing artifacts (e.g., data and services) by leveraging and extending existing model-driven software engineering techniques. As a proof-of-concept, we validated our ideas on the Android and iOS platform. However, there are many open issues (e.g., native vs. web) when building mobile applications that requires further exploration of other best practices in software engineering, which can evolve as a new discipline of its own.

[1] Instead the *checkbox* functionality can be captured via a *switch* in iOS

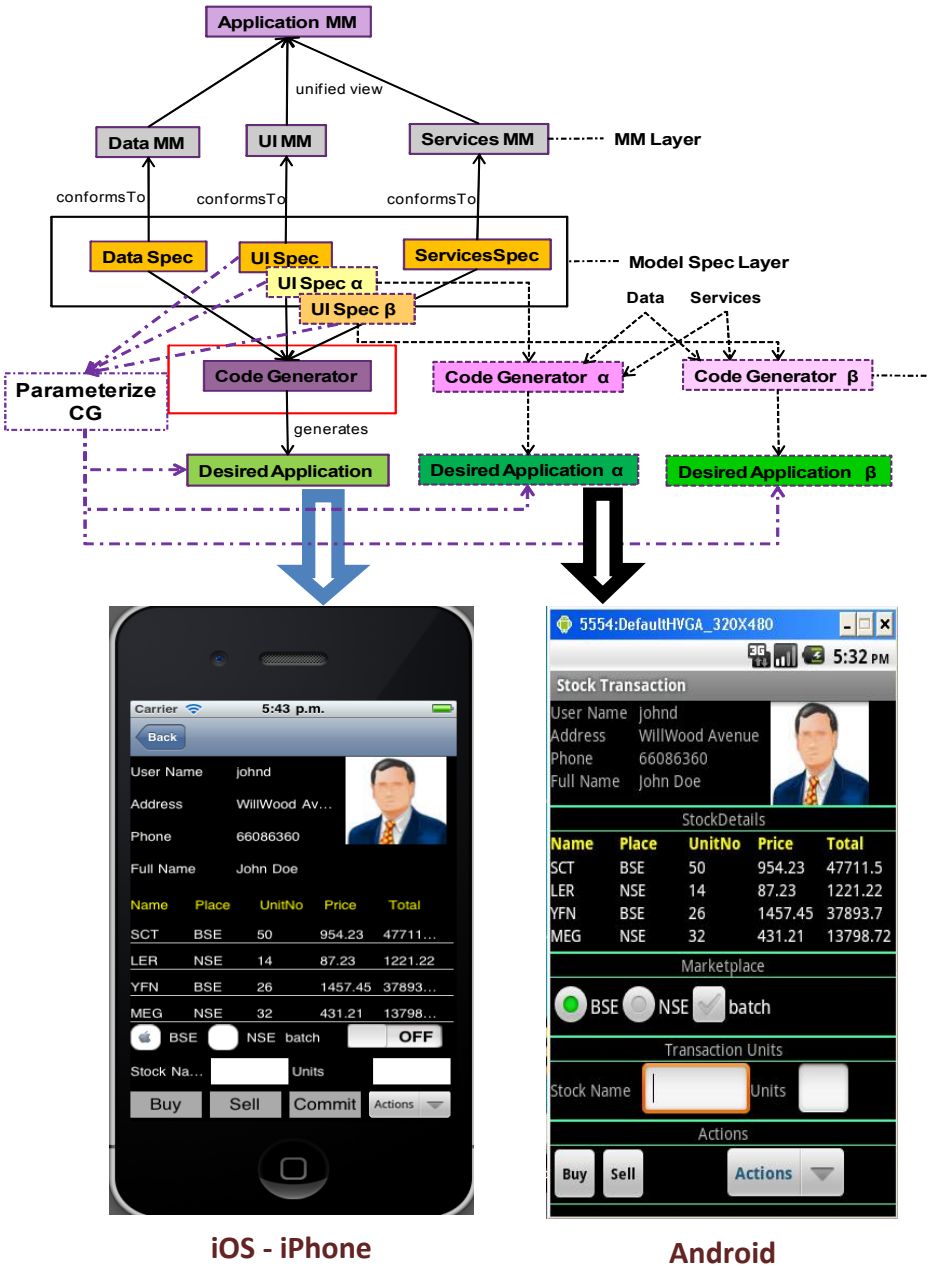

Fig 3: Mobile-Enabling Enterprise Business Apps

References

1. Tor-Morten Grønli, Jarle Hansen, Gheorghita Ghinea "*Android vs Windows Mobile vs Java ME: a comparative study of mobile development environments*", proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments , June 2010.
2. David Szczesny, Sebastian Hessel, Felix Bruns, Attila Bilgic, "*On-the-fly hardware acceleration for protocol stack processing in next generation mobile devices*", proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis, October 2009.
3. Michalis Zarifopoulos, Anastasios A. Economides, "*Evaluating mobile banking portals*", International Journal of Mobile Communications, January 2009, Volume 7 Issue 1.
4. Key Predictions for IT Organizations and Users in 2010 and Beyond, Gartner Report, January 2010, http://www.gartner.com/it/page.jsp?id=1278413.
5. Vinay Kulkarni, R. Venkatesh and Sreedhar Reddy. *Generating enterprise applications from models*. Object-Oriented Information Systems, 2002, LNCS 2426, pp. 270-279.

System Design and Software Engineering Challenges in Building an Android Application: a Case Study

Rawan Tarek Khalil and Ala' Fathi Khalifeh, German University in Cairo, Egypt
rawan.khalil@student.guc.edu.eg, alaa.khalifeh@guc.edu.eg

Abstract— With the tremendous increase in the number of mobile phone users. The demand on having mobile applications that provide solutions to many of the **people's** daily life aspects increases too. However, designing these applications is different from the typical desktop applications and imposes many software engineering challenges that need to be taken into consideration. In this paper, we present the system design and the software engineering challenges encountered while implementing an Android mobile application that provides a voiced based text messaging functionalities for people who tends to text while driving their cars.

Index Terms— Voice recognition, Android mobile development, automatic text messaging, voice based software development, mobile phone software engineering.

I. INTRODUCTION

With the way that technology has revolutionized communication methods, mobile phones have become an indispensable part of keeping up with this fast paced world. Mobile phones have provided an on-the-go easy, fast and efficient method of communication and have opened up a lot of possibility for enhancing user experience with the digital world. However, this widespread adoption of these devices has stimulated software architects to create new applications and solutions that opened up a new horizon for the software engineering community and imposed new challenges and limitation that did not exist in the personal computer applications.

People are always looking for the easiest, most user friendly and most intuitive method of communicating through their devices and there is probably no easier way to achieve that than talking to your phone. Speech is one of the most intuitive ways of interaction since it is the natural method of communication between people. Thus, speech recognition over mobile phones is rapidly becoming a must have feature in any device. For people on the go, for example while driving or walking, it is quite difficult and not to mention very dangerous to use mobile phones without any kinds of aids such as wireless Bluetooth earpieces and other accessories that make it hands free. According to [1], a shocking 16,000 deaths were caused by texting while driving in the US between 2001 and 2007 and it is on the rise every year. A solution to a problem would be provided through the use of Automatic Speech Recognition (ASR). Through ASR software, drivers will never have to take their eyes off the road, their hands off the wheel or their minds off their safety. They can respond to text messages by simply speaking their reply. The main aim of this paper to present the software design methodology and steps needed to design a voiced based text messaging application over Android mobile phones that try to address the aforementioned problem, with some emphasis on the software and design challenges faced during the implementation, which is of particular interest to the Android mobile developers' community.

The rest of the paper is divided into three further sections. Section II describes the application developed and its architecture. Section III presents some of the software engineering challenges faced during the implementation and how they were addressed. Finally, Section IV summarizes the paper.

II. SYSTEM DESIGN AND ARCHITECTURE

In this section, we provide a detailed description of the application, its architecture, and all the modules and components of the application.

A. Application Description

The main aim of the application is to enable users, specifically drivers, to use their mobile phones on the road without having to resort to physical contact with the phone. This approach for interaction with the device will greatly reduce the risk of accidents caused by distracted driving. The application focuses on the feature of text messaging or Short Message Service (SMS). When a mobile phone user receives a text message while driving, he is either not going to reply until they stop or they will have to pick up the phone to read and/or reply to the message. To provide a solution to that problem, we created this application that combines the features of text messaging with those of speech recognition and speech synthesis. Instead of having the user to type in the reply using the keyboard, the user could use the speech recognition feature and reply which is then automatically translated by the recognition engine. The application is based on the idea of having a set of keywords that correspond to longer stored messages. This means that each user has a list of words stored on their phone and each keyword is a sort of abbreviation for a longer text message. For example, the user defined keyword "later" could stand for "I'm busy right now. I will call you later". The user creates his/her own set of custom keywords and messages depending on their preferences and the most common situations that could be encountered while driving. The keywords are set to be exactly one word long in order to make it easier and faster while driving to just say a keyword instead of a whole sentence. Moreover, this improves the chances of accurate recognition rather than using long complex sentences.

When the user receives a text message, by using speech synthesis the system will ask the user if they would like to reply. If the user replies with "Yes", they will be prompted for the keyword they would like to send. Once the keyword is spoken by the user, if the recognition is correct, the message associated with that keyword is retrieved and used as the

message body for the reply. The user will then confirm the system to send the message. This is the core part of the application that uses all the key components. There is also a component for viewing the inbox and a component for viewing and editing keywords.

B. Application Architecture and Modules

Figure 1 shows the block diagram for the system architecture. There are three modules in the system; the keywords module, the messaging module and the dialogue module. The diagram shows the main Android packages used in each module. The boxes shaded in green correspond to Android packages [2] while the boxes shaded in blue correspond to custom classes. Below each box there is a note that contains all the classes used within each package. For the custom classes, the note boxes are also custom classes in the same module that are related to the main class. The boxes that exist outside any module like the Activity Manager are general items that were used in all modules.

1) Keywords Module: The keywords module is the component of the system that is responsible for managing the keywords. There are six main functionalities in this module. A user can display a list of all keywords, add, show, edit, delete or send a keyword. It is based on a simple Create, Render, Update and Delete (CRUD) system with a single two-column table database where one table entry corresponds to a keyword and its longer message. The user can see the list of keywords by selecting" Keywords" from the main screen. Once inside the keyword list, the user can then choose to view a specific keyword, add a new one or search for an existing keyword. If the user clicks on a certain keyword, he/she will be directed to the view or edit with the keyword and its message

2) Messaging Module: The messages module is a regular text messaging system. Using this module, users can view their inbox, send messages and reply to received messages. When users wish to send a message or reply to a message, they can choose from the list of stored keywords. Once they select the required keyword from the list, they will be directed to the show view of that keyword. They can then choose "Send" from the menu and will then be directed to the phone's text messaging application where they can choose a contact to send the message to. The message body is automatically filled out using the stored message from the chosen keyword and the user can edit it if he/she wishes to. In addition to the selection from the list, users can send messages by using speech recognition. When a message is received, they can reply to it by saying the keyword and the speech recognition will process the input speech and automatically send the message after confirmation from the user.

3) System-User Dialog Module: In order to make the application truly serve its purpose of eliminating physical and visual contact with the phone while driving, we combined the features of speech recognition and speech synthesis. Through these technologies, we were able to create a dialog between the user and the system on the event of receiving a message. This dialog is aimed at guiding the user to replying to the message without the need to look at the phone. When a message is received, the system informs the user through speech synthesis that he/she has received a new message from certain contact. The system then asks the user if he/she would like to hear the message. The system will then start the speech recognition and wait for the user to respond. If there is no response from the user, the system will repeat the request two more times and if there is still no response, it will exit on the grounds that the user is unavailable. The system will also exit in the case that the user replies with "No". However, if the reply is "Yes", then the system will read out the received message. After that, the system asks the user if he/she would like to reply or hear the message again. If the user replies with "Again", then the message is replayed and the dialog sequence resets from that point. If the user says "No", then the system will exit only after confirmation from the user. If the user says "Yes", then the system will prompt the user for the keyword they would like to use as the reply to the message. If the keyword spoken by the user was found in the database, then the user is asked to confirm that the recognition was correct. If yes, then the system proceeds to send the message and exit. However, if the user says "No" or if the recognition returned no result, then the system will ask the user to say the keyword again. If the system fails to find the keyword three times, the system then exits. At any given point in the speech recognition process, if the user says the word "Quit" the system will automatically exit.

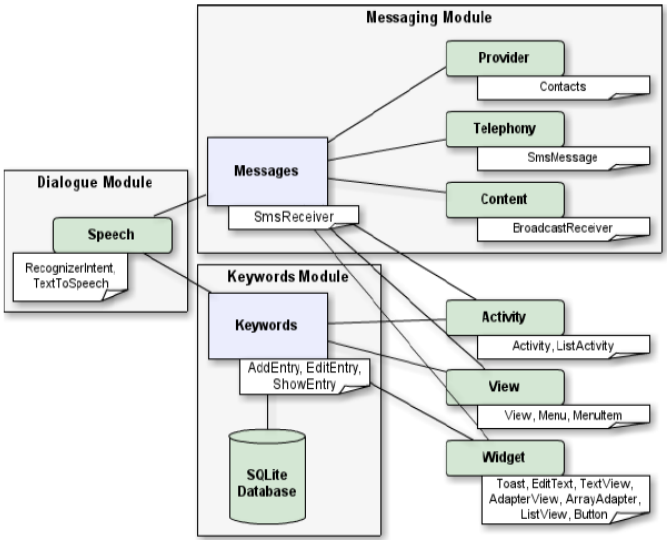

Figure 1. Application Architecture Block Diagram

III. SOFTWAR ENGINEERING CHALLENGES

In this section, we disucss some of the software challeneges specifict to the mobile device environment that we have faced during the system design and implementation.

Challenge 1: Mobile Hardware and Software Diversity: The diversity that now exists between software and hardware platforms for mobile devices is undeniable. It poses a great

challenge in choosing between them or trying to develop something that can work the same way across multiple platforms. Having to make this choice can be a difficult trade-off. Choosing a specific platform will limit the developer to the capabilities and features of this platform. Even within one platform, there could be several different versions that can make applications behave differently from one device to another. Developers must always try to make their applications work smoothly and as expected regardless of the device. Another significant problem is that it would limit the user base of the application to those that own supporting devices. However, sometimes limiting the user base is not necessarily a disadvantage. It depends on what the application needs and who it is targeted at. These problems can still exist with regular Software Engineering and not only Mobile Software Engineering. However, the problem is more pronounced and accounted for in mobile development since the scales of growth are now tipping into the direction of mobile devices over PCs. Moreover, most of the applications developed over PCs are usually web applications that are generally platform independent. In our application, we chose to develop on the Android platform. The reason for this selection is that the popularity of Android is increasing rapidly that it has now become the market leader. Furthermore, it is also open-source which gives the advantage of more developer collaboration and insight into the details. In addition, cross-platform adjustments required between Android devices is a simple process where the developer just needs to handle some design issues regarding the UI and defining the features that the application needs to run, so that if this feature is not present on a certain device, it would not install the application to avoid any crashes. In choosing Android, there are specific best practices recommended by developers that point others into the direction of what works better. This means that there are certain methodologies and concepts related to Android that distinguish the development process from other platforms. Hence, the software engineering procedure can not be generalized for other platforms.

Challenge 2: Mobile OS Frequent Updates and Releases: A challenge that may arise during mobile development is the ability of the application developed to be altered to match updates. Mobile platforms are very frequently updated and new versions come out quite often. Developed applications must be easily upgraded to fit the requirements and adapt to the changes of newer and better software versions. This is helped by the fact that Android SDK [2] comes in the form of several components that can be used to build an application. If an update occurs, then only the relevant components that are used in the application would need to be modified to fit the new version. For that reason, when developing for mobile software, it is better to separate different components from each other so that the application can be easily maintained and upgraded. Once it is altered, newer releases can be published to the market or app store.

Challenge 3: Power, Processing and Storage Management: Another significant challenge that comes with mobile development is the lack of computational power required to perform complex operations. Mobile devices that exist so far have very limited computational power when compared to PCs. In our application, the use of voice recognition requires a huge amount of processing power in order to understand and analyze the speech input from the user. This is why Android voice recognition is done remotely on Google servers [3] rather than locally on the device. The amount of power required to perform speech recognition operations is too exhausting for a mobile device with the current hardware limitations. Furthermore, such heavy and complex operations will also drain the battery out too quickly. Developers must try to keep in mind the effect of the processing of the application and the services used by the application that will require a lot of battery power. However, one disadvantage of off- loading the mobile phone from the recognition functionality that it requires the mobile device to be connected to the Internet which may not be always possible. Another challenge is related to the limited storage space on mobile devices. In order to perform voice recognition, the application would require that there be a huge dictionary of words that the user may say to match them up. This will require a lot of valuable space. This space is not an issue on PCs, but for mobile devices, space consumption must be minimized to fit in all the other applications and processes required by the device. This is also helped by the recognition engine actually being on Google servers. The only space consumption required by the application is a very light and a basic database managed by SQLite engine that consists of only one table of keywords and messages.

IV. CONCLUSION

In this paper, we presented an Android mobile based voice application with an emphasis on the design aspects, and the software engineering practices related to the Android mobile platform. Utilizing our application, the driver can respond to the received text messages while driving by issuing a voice commands to his mobile phones. Some of the software engineering challenges encountered while building the application were the fact that designing a mobile application is to a big extent vendor and platform dependent, this is why we have chosen Android platform as it is implemented in a large number of mobile phones from different vendors. Another challenge was related to the frequent updates and releases of the mobile OS which was addressed by making the application modular and upgradable. Finally, one should bear in mind the limited computational and storage capabilities of the mobile phone which were taken into account in our application by off-loading the mobile device from the recognition process and instead, relying on the remote Google servers.

REFERENCES

[1] F. Wilson and J. Stimpson, ”Trends in Fatalities from Distracted Driving in the United States, 1999 to 2008.” Am J Public Health, 100.11 (2010):2213-2219.
[2] Android SDK: http://developer.android.com/sdk/
[3] Google Mobile. http://www.google.com/mobile/voice-actions/.

Apps vs. Open Web: The Battle of the Decade

Tommi Mikkonen
Department of Software Systems
Tampere University of Technology
Tampere, Finland
tommi.mikkonen@tut.fi

Antero Taivalsaari
Advanced Development & Technology
Nokia Corporation
Tampere, Finland
antero.taivalsaari@nokia.com

Abstract—Today, both desktop and mobile software systems are usually built to leverage resources available on the World Wide Web. However, in recent years desktop and mobile software have evolved in different directions. On desktop computers, the most popular application for accessing content and applications on the Web is the web browser. In mobile devices, in contrast, the majority of web content is consumed via custom-built native web apps. This divergence will not continue indefinitely. We anticipate that **in the 2010's** we will witness a major battle between two types of technologies: (1) native web apps and (2) Open Web applications that run in a web browser or some other standards-compliant web runtime **environment. This "***Battle of the Decade***"** will determine the future of the software industry – as well as the future of software engineering research – for years to come.

Keywords–web applications, web programming, web-based software development, software engineering, web engineering, open web

I. INTRODUCTION

Although the World Wide Web has existed less than twenty years, the Web has already transformed our lives in countless ways. These days, everyday artifacts and services such as documents, photos, music, videos and newspapers are widely available on the Web. Online banking and stock trading have become commonplace. Various documents that used to be difficult to access, such as municipal zoning documents, government budget documents or tax records, are now readily available on the Web. Entire industries such as banking, financial services, electronics and book retailing, photography, and music distribution have undergone dramatic transformations. Web-based services such as Facebook and Twitter have altered the meaning of social life. The Web is even having a profound impact on politics and democracy, shaping the future of nations all over the planet.

The World Wide Web has also had a considerable impact on the software industry. These days, both desktop and mobile software systems are usually built to leverage resources available on the Web, with the objective that the same content can be accessed effortlessly from different types of terminals. However, in recent years desktop and mobile software systems have evolved in rather different directions. On desktop computers, the most popular application for accessing content and applications on the Web is the web browser. In mobile devices, in contrast, the majority of web content today is consumed via custom-built native web applications, or "apps" for short.

In this paper, we anticipate that in the 2010's we will witness a major battle between two types of technologies: (1) native web apps and (2) Open Web applications that run in a web browser or some other standards-compliant web runtime environment. The former approach implies the use of binary software and traditional software engineering practices, while the latter approach implies that conventional software engineering methods and practices will be replaced by technologies created for web development. This "*Battle of the Decade*", as we call it, will determine not only the future of the software industry, but the future of software engineering research as well.

This paper builds on a number of earlier papers [4, 5, 6, 7, 11, 12, 13, 14]. Many of the topics in this short paper have been covered more extensively in those earlier papers.

The rest of this paper is structured as follows. In Section II we provide a brief discussion on the evolution of the Web as a software platform, and then focus on the ongoing battle between native apps and Open Web applications in Section III. In Section IV we outline the research challenges that arise from the two divergent paths. In Section V we draw some final conclusions.

II. EVOLUTION OF THE WEB AS A SOFTWARE PLATFORM

Over the past twenty years, the World Wide Web has evolved from a document sharing system to a massively popular, general purpose application and content distribution environment – in short, the most powerful information distribution environment in the history of humankind. This evolution has taken place in a number of evolutionary phases or eras [14]. Note that here we intentionally focus on the evolution of the Web *as a software platform*. When viewed from other angles – e.g., from the viewpoint of online banking or music or video distribution – the history of the Web would look somewhat different.

In the first era – *the Web as a document environment* – the programming capabilities of the Web were very limited, reflecting the origins of the Web as a document sharing and distribution environment. In the second era – *the Web as an application environment* – the software development capabilities of the Web started emerging, with different technologies competing with each other vigorously. In the third era that is unfolding currently – *the Web as the application environment* – we believe that the landscape of the software industry will change dramatically, as the balance shifts irrevocably from binary end user software to web-based software. Note that these three eras are by no

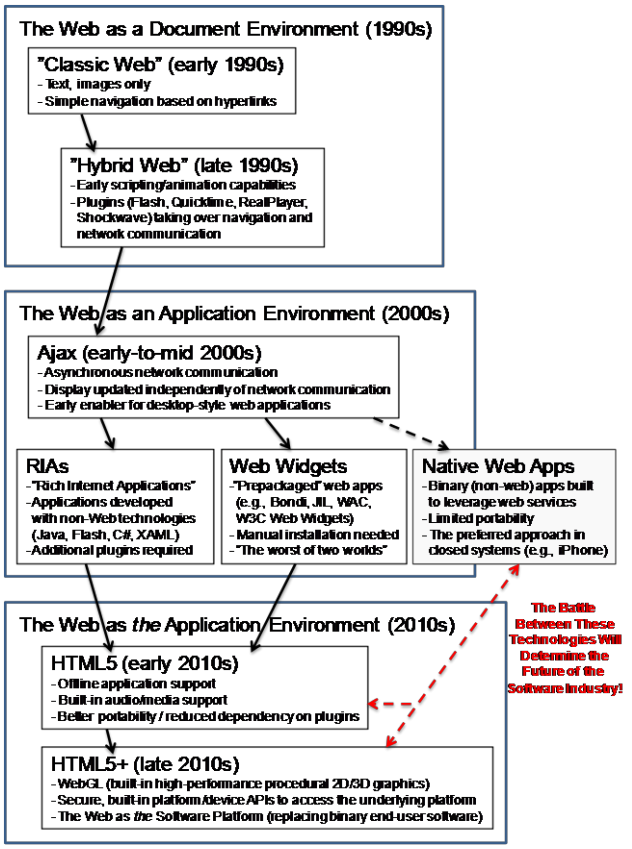

Figure 1. Evolution of the Web as a Software Platform
(for high-quality image, see http://lively.cs.tut.fi/WebEvolution.png)

means mutually exclusive. Rather, web pages and applications representing all three eras coexist on the Web today. A visual summary of the different eras is provided in Figure 1. We will ignore many of the details in this paper, since they are not relevant for the main theme of the paper.

III. The Battle of the Decade

The key point about the evolution of the Web presented in Figure 1 is the current, ongoing battle between native web apps and HTML5-based Open Web applications.

A. Native Apps

Custom-built native apps have become one of the dominant ways people use web services. For instance, on Apple's highly successful iPhone and iPad devices, as well as on Google's Android devices, the users typically access Facebook, Twitter, and many other popular web services using custom-built native apps rather than with the web browser. Such native apps are not really web applications at all; however, they use the same network protocols to access the backend services as the web browser does.

There are good reasons for native web apps to exist. While the underlying needs to communicate and access information are the same in desktop and mobile environments, the way people consume content and use applications with different types of terminals and devices are fundamentally different. In the mobile space, the time span of the users' actions is usually significantly shorter than in the desktop space; the users wish to perform rapid, focused actions instead of long-lasting sessions; actions must be simple yet focused, and they must accomplished with ease, using only a minimal number of keystrokes or finger presses, often while the user is walking, driving a car or is somehow otherwise distracted by other activities. The different usage modalities and smaller screen sizes have a significant impact on application design; generic web pages geared towards laptop or desktop computer users are not usually ideal for mobile use. In addition, performance issues or network connectivity issues can make web applications nearly unusable in mobile devices. The conventional web browser simply was not designed for such use.

By using the native graphics libraries, the look-and-feel of apps can be customized specifically to the needs of the application and the device; the applications can also leverage device-specific features much more comprehensively than a pure web application could. The downside of such apps is that they are *strictly platform-specific*. Apps developed for the iPhone run only on Apple devices, so several different implementations – composed with different platform-specific tools – are needed if the app is to run also on Android, Blackberry, Symbian, or other commonly used target platforms [15]. In many cases a separate app is needed for each of the different versions of the target device. Such fragmentation is what effectively killed Sun's (now Oracle's) once highly successful Java ME platform [10].

Another source of fragmentation is that different apps, developed by different parties, commonly assume different ways of interaction. For instance, gestures that work in a certain fashion in one application may imply totally different functions in other applications. This can be confusing for the user, and in the end lead to additional requirements on how applications should be defined, together with associated style guides and so forth.

Finally, unlike pure web applications, a native app requires conventional installation. The user must usually download the application binary from a specific location, such as Apple's App Store (*http://store.apple.com/*). In order to introduce new features, the user must typically download and install a totally new version or upgrade the application explicitly by device-specific means. This is clumsy and inconvenient for the user, e.g., since the application or the entire device may be partially unavailable while the download and upgrade is in progress.

B. Open Web

Following the Open Web principles laid out in the *Mozilla Manifesto* [8], web applications should be built on technologies that are open, accessible and as interoperable as possible, and should run in a standards compatible web browser without plugins, extensions or custom runtimes. In December 2010, Tim Berners-Lee – the inventor and founder of the World Wide Web – published an article in which he called the current trend towards custom-built native web apps "disturbing", because that trend is dividing information into separate content silos that are isolated from each other [2]. Such content is off the Open Web, and usually under the

control of an individual company. Typically, you cannot bookmark, tweet or e-mail a link to such a page using a standard browser. Rather, you must explicitly download, install and use (and later upgrade) a vendor-specific app from a vendor-specific app store for each device platform in order to access such content.

Open Web applications have various benefits. For instance, they require no installation or manual upgrades, and they can be deployed instantly worldwide. A web application published in Tampere (Finland), say, is instantly and equally available in Tallahassee (Florida, USA), Tandragee (Ireland) or Taree (New South Wales, Australia) without explicit installation. The Open Web principles will allow application development and instant worldwide deployment without middlemen or distributors. Conventional binary applications are at a major disadvantage when compared to web-based software that can be deployed instantly across the planet.

So far, a number of obstacles have hindered the development and deployment of full-fledged, truly interactive web applications. The obstacles have been especially apparent in the mobile device space. We have analyzed the problems in earlier papers [4, 5, 6, 12]. However, new standards such as HTML5 and WebGL will eliminate many of the limitations in this area.

The forthcoming *HTML5* standard [16] complements the capabilities of the existing HTML standards with numerous new features. Although HTML5 is a general-purpose web standard, many of the new features are aimed squarely at making the Web a better place for desktop-style web applications. Examples of features that support desktop-style applications include offline applications that can be run even when an active network connection is not available (http://www.w3.org/TR/offlinewebapps/), a simple storage mechanism that behaves like a simple key-value database, allowing textual data to be stored locally in the computer/device, Canvas API that provides a 2D drawing canvas for procedural, interactive graphics, and built-in audio and video support.

WebGL (http://www.khronos.org/webgl/) [3] is a cross-platform web standard for hardware accelerated 3D graphics API developed by Mozilla (http://www.mozilla.org) and Khronos Group (http://www.khronos.org/), and a consortium of additional companies including Apple, Google and Opera. The main feature that WebGL brings to the Web is the ability to display 3D graphics natively in the web browser without any plug-in components. Unlike with earlier technologies such as Flash, O3D, VRML and X3D, with WebGL the 3D capabilities are integrated directly in the web browser, meaning that 3D content can run smoothly and portably in any standards-compliant browser. The possibility to display 3D graphics natively in a web browser is one of the most exciting things happening on the Web recently.

While HTML5 and the related W3C standard activities play a critical role in turning the Web into a serious application platform, it is important to note that the feature set offered by an HTML5-compliant web browser is still somewhat incomplete for real-world applications. As depicted in Figure 1, our prediction is that another major round of standardization will be necessary in mid-to-late 2010s to establish a more complete web application platform. We refer to such standard work informally as "*HTML5+*", that is, the next major version/successor of the HTML5 Specification. A critical goal in that work will be to more comprehensively "virtualize" the underlying operating system and device capabilities, as well as ensure that the necessary security mechanisms are in place to access the platform and device capabilities securely.

IV. Impact on the Software Industry and Software Engineering Research

The document-oriented origins of the Web have led to an *impedance mismatch* between web development and conventional software engineering. In this section we take a brief look at this impedance mismatch and its implications for the software industry and software engineering research.

A. Web vs. Conventional Software Development: The Impedance Mismatch

As we have discussed in earlier papers, a historical impedance mismatch exists between web development and software engineering. This impedance mismatch reflects the fact that the World Wide Web was originally designed to be a document distribution environment – not a software platform. The differences are highlighted in Figure 2.

Web Development	Conventional SW Development
- Documents	- Applications
- Page / form oriented interaction	- Direct manipulation
- Managed graphics, static layout	- Directly drawn, dynamic graphics
- Instant worldwide deployment	- Conventional deployment
- Source code and text favored	- Binary representations favored
- Development based mostly on conventions and "folklore"	- Development based on established engineering principles
- Informal development practices	- More formal development
- Target environment not designed for applications	- Target environment specifically intended for applications
- Tool-driven development approach	- A wide variety of development approaches available

Figure 2. Impedance Mismatch Between Web Development and Conventional Software Development

In the remaining parts of the paper, we will consider the two divergent paths that the evolution of the Web may take as a result of the Battle of the Decade. The implications for the software industry and software engineering research are entirely different depending whether the balance tilts towards native apps or the Open Web.

B. Scenario 1: Native Apps Will Dominate

Many people seem to take it for granted that especially in the mobile industry native apps will continue to dominate. For instance, in a September 2010 Wired magazine article Chris Anderson and Michael Wolff claimed that the Web is already dead [1], because for the vast majority of web services such as e-mail, news, Facebook and Twitter, users will prefer custom-built native applications (e.g., Flipboard for iPad) over open, unfettered web browser access.

The success of native apps is not entirely unexpected. Native apps enjoy considerable success partly because of commercial reasons (e.g., because it tends to be easier to monetize closed rather than open platforms) and partly because of technical reasons (e.g., because it is easier to define new APIs and optimize overall system behavior in world in which the platform is owned and controlled by a single vendor.

Superficially, from the viewpoint of software engineering, the native apps scenario is business as usual. Since the development model in this scenario revolves around the creation of rather conventional binary applications that are written, installed and run in a well-known fashion, existing design, integration and testing practices and methods can be used without major changes.

However, under the surface there are numerous things that need attention. To begin with, mobile devices are subject to significantly more variations and fragmentation than conventional desktop computers. For instance, screen size differences, different interaction and input mechanisms, memory and processing power limitations/differences and intermittent network connections create additional challenges for developers. In the area of Java ME development – the once dominant mobile application platform – some game companies reported that they had to create over a thousand different variants of their applications for different devices!

These days, the mobile industry seems to be headed to an equilibrium in which two or three native platforms will dominate the industry. The companies controlling those platforms place a lot stricter restrictions on the device capabilities than the Java ME specifications ever did. Nevertheless, the application developers will still have to create a large number of variants of their applications if they expect their applications to be available on all the major platforms, devices and countries; even if the developer is targeting only one major platform such as Apple's iPhone, internationalization and localization may still require effort.

In general, the successful creation of commercial native web apps places a lot of requirements on product family management. In order to offer an attractive app portfolio that covers all the different platforms, tools for managing fragmentation in massive scale are needed. Those tools must be able to provide cross-platform support that enables the use of the same code in different platforms, and is capable of recognizing and handling the micro-level fragmentation issues (bugs and "features") between different devices that use the same platform. The tools must also be able to take into account the different installation practices for different target platforms. For instance, for Apple's iPhone and iPad devices, application installation can only take place via Apple's Web Store.

The topics discussed above are just a tip of the iceberg for a proper research agenda for Scenario 1.

C. Scenario 2: Open Web Will Dominate

The starting point for Scenario 2 is that the transition towards web-based software development will continue and will eventually have a profound impact not only for desktop software but mobile software development as well.

The victory of Open Web applications is by no means guaranteed, though. There are still numerous issues that plague the development of web applications, and for mobile devices especially. In our earlier papers, we have divided those problems broadly into the following categories:

(1) software engineering principle violations,
(2) usability and user interaction issues,
(3) networking and security issues,
(4) browser interoperability and compatibility issues,
(5) development style and testing issues,
(6) deployment model issues, and
(7) performance issues.

We will not revisit all the categories in this brief paper. Rather, we highlight a number of topics that we believe should be high on the research agenda for Scenario 2.

First, the transition from binary applications to pure web applications will result in a shift away from static programming languages such as C, C++ or C# towards dynamic programming languages such as JavaScript, PHP or Python [9]. Since mainstream software developers are often unaware of the fundamental development style differences between static and dynamic programming languages, there is a need for education in this area. Developers need to be educated about the evolutionary, exploratory programming style associated with dynamic languages, as well as agile development methods and techniques that are available for facilitating such development.

Second, the software deployment practices for web applications are entirely different from conventional binary software. Web applications are distributed primarily in the form of source code, not binaries. Any application updates that are posted on the Web are immediately accessible to anybody anywhere on the planet. This "instant gratification" dimension will revolutionize the deployment and distribution of software applications, and will enable "nano releases", i.e., software releases that may occur multiple times per day or even every few minutes. For instance, recently Netflix (http://www.netflix.com/) reported that they commonly publish updates to their web applications up to six times per day! One of the main challenges in the deployment area is to define a model that addresses the fundamental changes in the nature of applications: applications that remain "always on", the ever-shortening nano release cycles, and the "perpetual beta syndrome", i.e., applications that will stay in continuous development mode indefinitely [11].

Third, in the testing area there is an increased need for code coverage testing methods to ensure that all the parts and execution paths of the applications are tested appropriately. Since web applications consist of pieces that are loaded dynamically without any static compilation, type checking or linking, it is quite possible for significant pieces of the applications to be missing at runtime. This feature, when combined with the lack of well-defined interfaces and general fragility that characterize web-based software [6, 12], leads to many interesting research topics and challenges, especially when developing mashups and mashware, i.e., software that dynamically combines content and components published in different sites all over the world.

In general, many of the development and deployment practices that are common in web-based software development go against the grain or even obliterate many of the established software engineering principles. So far, there has not been enough discourse between the software engineering and web engineering communities; this is definitely an area for future improvement.

V. CONCLUSIONS

In this paper, we have argued that the ongoing "*Battle of the Decade*" between native web apps and HTML5-based Open Web applications will determine the future of the software industry and software engineering research. We started the paper by summarizing the evolution of the Web as a software platform, followed by an overview of native web apps vs. Open Web applications that run in a web browser or some other standards-compliant web runtime environment. We then presented two alternative scenarios for the future of the industry based on the possible outcomes of the battle, as well as highlighted interesting areas for future research.

REFERENCES

[1] C. Anderson, and M. Wolff, "The Web is Dead: Long Live the Internet," *Wired*, Sep 2010, pp.118-127, 164-166.

[2] T. Berners-Lee, "Long Live the Web: a Call for Continued Open Standards and Neutrality," *Scientific American*, vol 303, nr 4 (Dec), 2010, pp.56-61.

[3] Khronos Group, *WebGL Specification*, Editor's Draft, August 8, 2011. URL: http://www.khronos.org/registry/webgl/specs/latest/

[4] T. Mikkonen and A. Taivalsaari, "Web Applications – Spagetti Code for the 21st Century," Proc. 6th ACIS International Conference on Software Engineering Research, Management, and Applications (SERA'08), IEEE Computer Society Press, 2008, pp. 319-328.

[5] T. Mikkonen and A. Taivalsaari, "Creating a Mobile Web Application Platform: The Lively Kernel Experiences," Proc. 24th Annual ACM Symposium on Applied Computing (SAC'09), ACM Press, 2009, pp. 177-184.

[6] T. Mikkonen and A. Taivalsaari, "The Mashware Challenge: Bridging the Gap Between Web Development and Software Engineering," Proc. 2010 Workshop on Future of Software Engineering Research (FoSER'10), ACM Press, 2010, pp. 245-249.

[7] T. Mikkonen and A. Taivalsaari, "Reports of the Web's Death Are Greatly Exaggerated," *IEEE Computer*, vol 44, nr 5, 2011, pp.30-36.

[8] Mozilla, *The Mozilla Manifesto*, 2011. URL: http://www.mozilla.org/about/manifesto.en.html

[9] L. D. Paulson, "Developers Shift to Dynamic Programming languages," *IEEE Computer*, vol 40, nr 2 (Feb) 2007, pp. 12-15.

[10] R. Riggs, A. Taivalsaari, J. Van Peursem, J. Huopaniemi, M. Patel, and A. Uotila, *Programming Wireless Devices with the Java™ 2 Platform, Micro Edition* (2nd Edition). Addison-Wesley (Java Series), 2003.

[11] A. Taivalsaari, T. Mikkonen, D. Ingalls, and K. Palacz, "Web Browser as an Application Platform," Proc. 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'08), IEEE Computer Society, 2008, pp. 293-302.

[12] A. Taivalsaari and T. Mikkonen, "Mashups and Modularity: Towards Secure and Reusable Web Applications," Proc. 1st International Workshop on Social Software Engineering (SoSEA'08, L'Aquila, Italy), Department of Software Systems, Tampere University of Technology, Report 1, 2008, pp 21-28.

[13] A. Taivalsaari, T. Mikkonen, M. Anttonen, and A. Salminen, "The Death of Binary Software: End User Software Moves to the Web," Proc. 9th Conference on Creating, Connecting and Collaborating through Computing (C5'11), IEEE Computer Society, 2011, pp. 17-23.

[14] A. Taivalsaari and T. Mikkonen, "The Web as a Platform: The Saga Continues", Proc. 37th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'11), IEEE Computer Society, 2011, pp. 170-174.

[15] T. Wasserman, "Engineering Issues in Mobile Application Development," *Invited talk at MobiCase'10*, October 25-27, 2010, Santa Clara, California, USA.

[16] World Wide Web Consortium, *HTML5 Specification*, W3C Editor's Draft, September 10, 2011. URL: http://www.w3.org/TR/html5/

ABOUT THE AUTHORS

Dr. Tommi Mikkonen is a Professor of Computer Science at Tampere University of Technology, Finland. Tommi has pioneered the education of mobile software development in Finland, and he has arranged numerous courses on software engineering and mobile computing. Tommi's current research interests include cloud computing, web programming, embedded systems, and mashup development.

Dr. Antero Taivalsaari is a Distinguished Engineer at Nokia. Antero is best known for his seminal role in the design of the Java™ Platform, Micro Edition (Java ME platform) – one of the most successful commercial mobile software platforms in the world, with over three billion devices deployed so far. Since 2006, Antero's research has focused on web application technologies and web-based software development especially for mobile devices.

Together, Tommi and Antero lead the Lively Web Programming Research Team at Tampere University of Technology. For further information and a full list of research team publications, refer to http://lively.cs.tut.fi/.

Mobile Application Software Engineering: Challenges and Research Directions

Josh Dehlinger and Jeremy Dixon
Department of Computer and Information Sciences
Towson University
jdehlinger@towson.edu, jdixon6@students.towson.edu

ABSTRACT

The rapid proliferation and ubiquity of mobile, smart devices in the consumer market has forced the software engineering community to quickly adapt development approaches conscious of the novel capabilities of mobile applications. The combination of computing power, access to novel onboard sensors and ease of application transfer to market has made mobile devices the new computing platform for businesses and independent developers. However, the growth of this new computing platform has outpaced the software engineering work tailored to mobile application development. This position paper looks at four significant challenges to mobile application software engineering and provides a discussion of possible research directions, drawing from existing areas of software engineering, that should be further examined. Specifically, we examine the challenge of: 1) creating user interfaces accessible to differently-abled users; 2) handling the complexity of providing applications across multiple mobile platforms; 3) designing context-aware aware applications; and, 4) specifying requirements uncertainty.

1. INTRODUCTION

Smart, mobile devices (hereafter, mobile devices) are the fastest growing computing platform with an estimated 1.6 billion mobile device users by 2013 (compared to a current estimate of 2 billion PC users) [1]. This rapid proliferation of mobile devices over the last five years has dramatically altered the platform that is utilized for social, business, entertainment, gaming, productivity and marketing using software applications. Containing global positioning sensors, wireless connectivity, photo/video capabilities, built-in web browsers, voice recognition, among other sensors, mobile devices have enabled the development of mobile applications that can provide rich, highly-localized, context-aware content to users in handheld devices equipped with similar computational power as a standard PC [2]. Yet, these same novel features/sensors found in mobile devices present new challenges and requirements to application developers that are not found traditional software applications [3].

The combination of computing power, access to novel onboard sensors and the ease in which applications can be monetized and transferred to the marketplace has made mobile application the new *IT* computing platform for development. However, the rapid proliferation of mobile devices and applications has outpaced the software engineering approaches tailored to mobile application software engineering.

Traditional software engineering approaches may not directly apply in a mobile device context. First, mobile device user interfaces (UI) provide a new paradigm for new human-computer interaction sequences (e.g., multi-touch interfaces, QR code scanning, image recognition, augmented reality, etc.) that have not been previously explored in research and of which no established UI guidelines exist [2], [4]. Second, the divergent mobile platforms (e.g., iOS, Android, Windows 7, etc.), differing hardware makers for platforms (e.g., Android versions found on HTC, Google, Samsung) and mobile phone and tablet platforms (e.g., Apple's iPhone and iPad) have necessitated developers to make a series of the same application tailored for each type of device [3]. Third, the novelty of a truly mobile computing platform provides both unique opportunities and challenges [3]. For example, Roman, Picco and Murphy assert that "mobility represents a total meltdown of all the stability assumptions" made in software engineering [5].

In this position paper, we discuss how these three factors present four significant challenges to mobile application software engineering that are critical to enable the design and development of quality mobile application utilizing the capabilities provided by mobile device hardware and platforms.

2. MOBILE APPLICATION SOFTWARE ENGINEERING

Based on the three factors novel to mobile application development outlined in Section 1, we outline the following fundamental, unique challenges to the state-of-practice in mobile application software engineering:

- Creating Universal User Interfaces. There has been some preliminary research in creating a universal user interface for mobile devices (c.f., [2], [4]). Each mobile platform has a unique guide to address developer user interface requirements. The user interface guidelines have several overlapping themes.

  A significant consideration for mobile UI development relates to screen size and resolution. For example, Apple devices are limited to two sizes based

on the size of the iPhone and the iPad where as Windows 7, Android, and Blackberry provide screens of varying sizes and screen resolutions. As a result, UI design is difficult and mobile application developers must anticipate the targeted device(s).

Shneiderman's "8 Golden Rules of Interface Design" have been well received since their introduction [6]. However, these rules may not equally apply to mobile devices. Research by Gong and Tarasewich suggest that four of Shneiderman's guidelines readily translate to mobile devices, including: enabling frequent users to use shortcuts, offering informative feedback, designing dialogs to yield closure, and supporting internal locus of control. The remaining rules must be modified to be made applicable to mobile development [7].

As these challenges continue to evolve, further research should focus on streamlining application development efforts regardless of the mobile platform or device. Significant effort should be directed towards anticipating the diverse landscape of user capabilities, user interfaces and user input techniques.

- Enabling Software Reuse across Mobile Platforms. Mobile applications currently span several different operating system platforms (e.g., iOS, Android, Windows 7, etc.), different hardware makers (Apple, HTC, Samsung, Google, etc.), delivery methods (i.e., native application, mobile web application) and computing platforms (i.e., smartphone, tablet). Each of these options must be considered during mobile application development as they have a direct influence on the software requirements. Companies currently need to make a business decision to target a single mobile device platform with rich features, multiple platforms through a mobile website with less rich features or spend the resources necessary to broadly target the gamut of mobile devices with rich, native applications. If targeting a single platform, developers may decide to build a single application for all platforms at the risk of some functional inconsistencies or instead consider building multiple version targeting each hardware/computing platform [3][8].

  Within this development environment, many companies have separate development teams or separately contracted out the development efforts for different platforms (e.g., iOS and Android) essentially redoubling the software engineering effort needed for functionally similar mobile applications. Even when development is coordinated amongst development teams targeting different platforms, it is often in an ad hoc basis without a concerted effort to reduce the development time and cost through existing, reuse-conscious software engineering methodologies.

  Recent efforts in adapting HTML5 with tools like PhoneGap aim to reduce the development effort to produce nearly native applications across multiple platforms by rendering native applications interfaces through webviews [9]. However, this approach does not allow for rich features that have access to the mobile device's API and is a technological solution rather the desired software engineering approach to reuse early software engineering assets.

- Designing Context-Aware Mobile Applications. Mobile devices represent a dramatic departure from traditional computing platforms as they no longer represent a "static notion of context, where changes are absent, small or predictable" [5]. Rather, mobile devices are highly personalized and must continuously monitor its environment, thereby making mobile applications inherently context aware (collectively time-aware, location-aware, device-aware, etc.) [10], [11]. Mobile applications are now contextualizing proximity, location, weather, time, etc. to deliver hyper-specialized, dynamic, rich content to users through context-aware applications. Previously, web applications would often provide contextualized content based on time, detected location and language. However, the extent of context-awareness currently possible in mobile applications is beyond what software engineering approaches have encountered outside of agent-oriented software engineering [12]. The consideration of context-awareness as a first-class feature in mobile application software engineering is needed so that the requisite attention is paid by developers when analyzing these requirements resulting in better designed context-aware applications.

- Balancing Agility and Uncertainty in Requirements. While most mobile application developers utilize an agile approach or a nearly ad hoc approach, the growing demand for context-aware applications, competition amongst mobile applications and low tolerance by users for unstable and/or unresponsive mobile applications (even if free) necessitates a more semi-formal approach. This should be integrated into agile engineering to specify and analyze mobile application requirements. The dynamic, contextual nature of mobile application content (e.g., location-based applications) allows for situations in which the application's behavior may not be able to fully satisfy the specified functional and non-functional requirements thereby necessitating that the application be self-adaptive. In this scenario the software will then provide less rich content satisfying less stringent requirements. For some mobile applications, this may arise if, as determined in the requirements, it is better for the application to run continuously and, when necessary, to autonomously modify its behavior and provide reduced functionality rather than provide no functionality at all. For example, in a location-based

application several factors (e.g., low battery, GPS sensor disabled, etc.) may affect the granularity and recentness of its content. In some location-based applications, it may be better to provide old content (i.e., content based on a previous location) rather than displaying an error message or risk slow or no response from the application.

Within mobile application software engineering, the need for an application to self-adapt, depending on context, has been constructed using ad hoc approaches. Yet, as mobile applications become more context-aware, self-adaptive requirements will need to be more formally integrated into agile development so that developers more rigorously consider the behavior of an application when its full requirements cannot be satisfied dynamically and how it can self-adapt to partially satisfy the requirements.

3. Research Directions for Mobile Application Software Engineering

This section builds off of the challenges outlined in Section 2 and provides sketches of future research directions in existing software engineering fields that can contribute to mobile application software engineering.

3.1 User Interfaces for the Differently-Abled

As development of mobile applications continues to expand, research and development regarding accessibility and utility for users who are differently-abled will become essential. Recent US Census data reports that approximately 15% of the United States population has at least one disability, including but not limited to sensory and physical limitations [13]. Yet, limited data exists to identify specific needs of this community in relation to mobile device application development and software engineering.

Some guidelines exist for modifications and development to assist those individuals with visual impairment (e.g., Apple's VoiceOver software for the iOS platform). These guidelines suggest the utilization of the VoiceOver software to help blind and low-vision users, which works as a screen reader and requires minimal additional information for most standard interfaces [14]. Development of specific applications for those individuals with other disabilities (e.g., physical and processing differences) has not yet been explored.

3.2 Mobile Application Software Product Lines

To support the reduction in cost in the development of functionally similar mobile applications across several platforms, mobile application software engineering must proactively make use of existing reuse-conscious software engineering approaches like software product line engineering (SPLE). SPLE supports reuse by developing a suite of applications sharing a common, manage set of requirements and is advantageous as it exploits the potential for reusability in the analysis and development [15]. A software product line a set of applications developed by a company that share a common set of core requirements yet differ amongst each other according to a set of variable requirements [15]. This approach can reduce time and cost needed in software engineering and "can arguably be viewed as the most successful approach to intra-organizational reuse of software" [16].

Weiss and Lai defined a two-phase SPLE approach as follows: the *domain engineering* phase defines the requirements (both common and variable) for the entire product line and the *application engineering* phase reuses these to develop specific applications within the product line [15]. The approach may be suitable to mobile application software engineering in that it would encourage developers to proactively focus on what the common requirements, design, resources, etc. to the development of a mobile application across different OS platforms (e.g., iOS, Android, etc.) or hardware platforms (e.g., HTC, Samsung, Google, etc. for the Android OS).

Integrating SPLE into mobile application software engineering encourages developers to assess the requirements for an application in a platform-independent manner and focus on what can be common across all versions of the application. It would also shift the mobile application software engineering process to develop application requirements upfront, rather than assigning the design and development to different, possibly independent development teams/contracts that may/may not coordinate in their efforts. Research efforts should look to how SPLE can be specifically tailored for mobile application software engineering to avoid duplicating early software engineering work and/or assets.

3.3 Context-Aware Applications

Context-awareness is novel feature and one of the primary factors driving the popularity of mobile applications [5], [11]. To support the design and development of context-aware applications, mobile application software engineering must incorporate context-aware software engineering approaches like those existing in agent-oriented software engineering (AOSE).

AOSE provides high-level abstractions, models and software engineering approaches for developing the autonomous software agents of a multi-agent system (MAS) [12]. One vital consideration for MAS is being context aware. Agents in a MAS must sense and react to its surrounding environment to be able to achieve its desired goals (i.e., functional requirements). This is increasingly the case with mobile applications.

Studying how some of the concepts/abstractions developed for AOSE can be utilized and/or adapted for mobile application software engineering may improve the design of context-aware applications and further mobile device innovation.

3.4 Self-Adaptive Requirements

Non-functional requirements are critical to mobile applications [3], and some mobile applications may need to dynamically self-adapt to provide reduced functionality. To better support the dynamism in mobile applications as a result of context-awareness and design for self-adaptation, mobile application software engineering should adapt existing self-adaptive systems requirement specification approaches like RELAX [17].

Whittle et al. proposed the requirements specification language RELAX as a medium of explicitly expressing environmental and behavioral uncertainty for the behavior of dynamically adaptive systems [18]. Within RELAX, requirements are partitioned to those that are invariant (i.e., requirements that must always be satisfied) and variant (i.e., requirements that may be partially satisfied) and then are specified in a structured natural language based on fuzzy logic and using fuzzy logic and using modal, temporal and ordinal operators. For each variant requirement, the RELAX process documents what environmental changes can affect the requirement and how the requirement can be partially satisfied. This approach extends the traditional *shall* requirement expression to also include keywords including *as early as possible, as close as possible to, eventually, as many as possible*, etc. to document the uncertainty and how the application can adapt in the face of uncertainty to still deliver some functionality.

Adapting RELAX into mobile application software engineering will direct developers to consider how an application could adapt when the environment or its behavior is non-optimal [18]. Integrated into an agile approach, it would provide better requirements structure and improve analysis and satisfaction of non-functional requirements in mobile applications when the environment/context changes.

4. Concluding Remarks

This paper briefly described four current challenges that we see for mobile application software engineering: designing universal UIs, developing for mobile application product lines, supporting context-aware applications and balancing agility with specifying requirements uncertainty. This paper asserts that mobile application software engineering research efforts need to focus on development approaches emphasizing UI design, proactive reuse at early software engineering phases, attention to context-awareness and sensitivity to specifying requirements to handle requirements uncertainty within the existing agile development approaches used for development applications. In addition, software engineering research needs to emphasize education initiatives in these four areas to ensure that these approaches are disseminated to those doing actual mobile application development.

5. References

[1] Gartner Group, "Gartner Says More than 1 Billion PCs In Use Worldwide and Headed to 2 Billion Units by 2014," 2008. [Online]. Available: http://www.gartner.com/it/page.jsp?id=703807. [Accessed: 11-Sep-2011].

[2] A. Oulasvirta, M. Wahlström, and K. Anders Ericsson, "What does it mean to be good at using a mobile device? An investigation of three levels of experience and skill," *International Journal of Human-Computer Studies*, vol. 69, no. 3, pp. 155-169, Mar. 2011.

[3] A. I. Wasserman, "Software engineering issues for mobile application development," in *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*, 2010, pp. 397-400.

[4] F. Balagtas-Fernandez, J. Forrai, and H. Hussmann, "Evaluation of user interface design and input methods for applications on mobile touch screen devices," *Human-Computer Interaction*, pp. 243–246, 2009.

[5] G. C. Roman, G. P. Picco, and A. L. Murphy, "Software engineering for mobility: a roadmap," in *Proc. of the Conf. on the Future of Software Engineering*, 2000, pp. 241–258.

[6] B. Shneiderman, "Designing the user interface," 1987.

[7] J. Gong and P. Tarasewich, "Guidelines for handheld mobile device interface design," in *Proceedings of DSI 2004 Annual Meeting*, 2004, pp. 3751–3756.

[8] B. Fling, *Mobile design and development*. O'Reilly, 2009.

[9] S. Allen, *Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile, and Android Development and Distribution*, 1st ed. Apress, 2010.

[10] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger, "Context-awareness on mobile devices - the hydrogen approach," in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, 2003.

[11] J. Dey, Anind K., Hakkila, "Context-Awareness and Mobile Devices," 2008.

[12] N. Jennings, "Agent-oriented software engineering," *Multi-Agent System Engineering*, pp. 1–7, 1999.

[13] M. Brault, "Disability status and the characteristics of people in group quarters: a brief analysis of disability prevalence among the civilian noninstitutionalized and total populations in the American community survey," *US Census Bureau*, 2008.

[14] Apple Inc., "iOS Human Interface Guidelines: Introduction." [Online]. Available: http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html. [Accessed: 13-Sep-2011].

[15] D. Weiss and C. Lai, "Software product line engineering: a family based software engineering process." Addison-Wesley, 1999.

[16] J. Bosch, "From software product lines to software ecosystems," in *Proceedings of the 13th International Software Product Line Conference*, 2009, pp. 111-119.

[17] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "RELAX: a language to address uncertainty in self-adaptive systems requirement," *Requirements Engineering*, vol. 15, no. 2, pp. 177-196, Mar. 2010.

XModel: an Unified Effort Towards the Development of High-Quality Mobile Applications

Érika Cota, Luigi Carro, Lucio Duarte, Leila Ribeiro, Flávio Wagner

PPGC - Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Po Box 15064 - Porto Alegre, RS, Brazil
{erika,carro,lmduarte,leila,flavio}inf.ufrgs.br

ABSTRACT

This paper proposes a model-based line of research and education for establishing new development approaches for mobile applications, where several non-trivial quality aspects of the product must be considered. We first detail our view of the main requirements for a mobile design methodology and discuss why traditional software engineering processes fail to address such requirements. In short, we believe the hierarchical view of a mobile system is not actually available to the software developer and this precludes productivity and overall quality of the resulting systems. We then present our view on how new design methodologies should tackle the existing challenges and actually provide design and implementation layers that can improve productivity and quality. The proposed line of action is based on the definition of appropriate high-level models and on multi-disciplinary knowledge, shifting the focus of education and research to topics that are currently marginal in the software engineer curriculum, such as optimization theory, model-based design and verification.

1 Mobile System Development Paradox

Mobile applications seem to converge to a generalization-customization paradox. To see that, let us consider the classic layered view of an electronic-based system, which includes a Business layer, a Software Development Kit (SDK) layer (development software), a Hardware-dependent-Software (HdS) layer and a Hardware layer (execution platform), where one layer is positioned above the next in the mentioned order, creating a hierarchical structure. Unfortunately, this hierarchical view of the execution and development of an embedded system is realistic only for a few classes of applications, namely, the ones with very specific requirements, usually having a single target function, such as automotive supporting systems, instrumentation, etc. Embedded systems targeted to the mass consumer market or, currently, mobile systems, have a more complex structure, as shown in Fig. 1. The underlying concept is the *variability* present at all levels. The reality that is not represented in this view is that hierarchy is indeed accomplished in the execution stack, but not in the development process, which is still "monolithic" in many aspects.

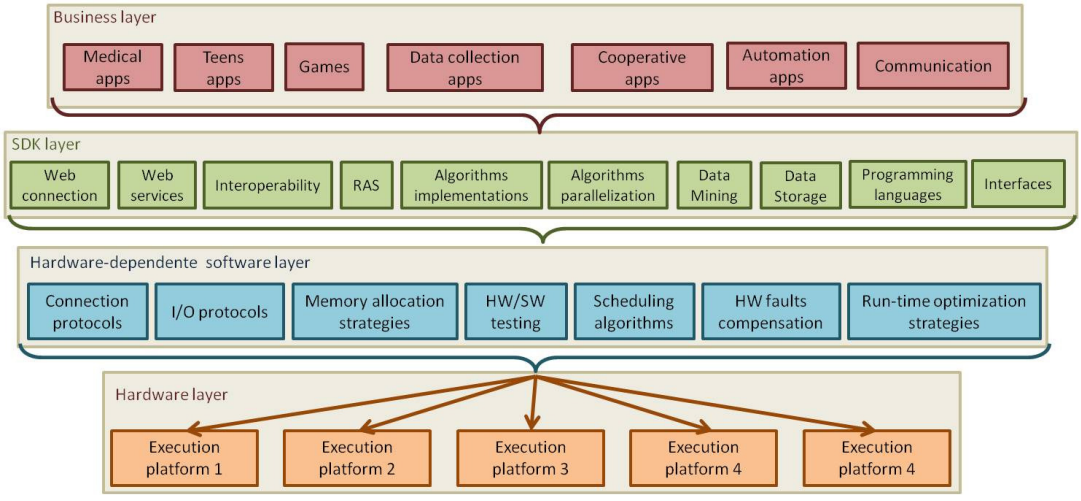

Figure 1: Embedded systems organization

In our view, *actual hierarchy* in the development process is the key to achieve high-quality mobile systems (for any definition and metrics of quality) but current practice of software engineering fails to support this development paradigm. In this paper we elaborate this view and present a possible research and education path towards the goal of a systematic, hierarchical, and flexible approach to mobile application development. Mobile applications are based on and driven by variability. Multiple execution platforms and all its variations (hardware layer) provide a broad range of computational power and development costs, thus allowing the use of an electronic-based system in a broad range of application fields (business layer). Similarly, computer science fields are in constant evolution towards the development of new technologies for the HdS layer (e.g., communication, optimization, and parallelization) and even more for the SDK layer (e.g., algorithms, and data manipulations). This variability at all levels potentially gives the system designer the necessary flexibility and adaptability to design a cost-effective solution that fits the specific application requirements and constraints.

On the other hand, programming within this huge design space can be cumbersome without levels of abstraction. Indeed, mobile software development is making use of abstraction layers. The hardware layer is accessible through development kits and devices drives provided by specific implementations of the HdS layer (different embedded operating systems and device drivers compiled to specific platforms). The concept of software platforms has also been used to leverage many of the challenges of the development of embedded software, which are normally related to the interaction with the lower layers. Software platforms, such as Android [1], make it easier for an ordinary programmer to access the resources of the hardware platform and interact with other applications or system resources as well. Cross-platform development environments, such as RhoMobile Rhodes [2] and PhoneGap [3], go one step further and help the developer deal with the hardware variability.

From the concept of design platforms (hardware and software) one can conclude that embedded systems tend to be based on generic solutions that are built to be customized at some point later in the development cycle. This approach is in line with the market trend of the mobile field where generic products are offered and further customized by end users according to their own needs. At this point, one can classify the mobile software designer in two groups: the *software platform provider* and the *application developer*. The first group is responsible for the implementation of the services provided by the HdS and SDK layers whereas the other group implements the final applications that will actually interact with an end user. Both groups develop mobile applications, but each one requires different backgrounds.

The application developer customizes a device by developing very specific applications and using the programming resources provided by the software platform. The concerns of an application developer should focus on the business and end-user logic, and not on the translation of this logic into, for instance, a power-efficient, low-level code that must execute over multiple target hardware. For this to be possible, the resources of the development platform should hide the details of the actual implementation and concentrate on the behavior or operations that actually represent the business model. As a consequence, the platform provider must focus on providing a development environment that can both capture the needs of the end user programmer and generate a high quality running code. This group should be concerned with the definition of mechanisms to automate the translation of the services provided by the HdS layer to the plethora of execution platforms available, and the adaptability of those services with respect to the constraints of the execution process (performance, memory, power-management, etc). This group must also find appropriate abstractions to create a high-level access for the resources at each level, in such a way that those can be used by an application developer that is not a computer science specialist. More than abstractions, the software platform must also take advantage of the constant new advances and variations of the lower-level layers to ensure levels of performance, reliability, availability, and overall system quality. For instance, services provided by the platform must be power-efficient, which can change according to the use of the service or even to the data. The platform must also be flexible to easily accommodate new resources or variations in the hardware and future application needs.

We advocate thus that software engineering for mobile systems should focus on the implementation of flexible and hierarchical features in the SDK and HdS layers, which is only

possible if the lower-level layers also provide the correct abstraction and information. Current practice, however, requires that each type of developer accumulates concerns and knowledge of different layers. This implies a longer learning curve for the use of a platform, and may preclude the implementation of high-quality software, since many decisions are taken without proper knowledge and information. Moreover, many decisions taken during the development might need to be adapted during execution or during the application lifetime. The current design model does not support this level of flexibility. Even higher-level platforms such as PhoneGap or Rhodes still assume the application developer has programming and computing knowledge to develop their application. Hence, it seems to us that the developers of the software platforms (HdS and SDK layers) is the one who needs a strong computing and engineering background, so that they can be able to provide an abstract framework of high quality for mobile customization. *Supporting mechanisms for the development of high quality embedded software development frameworks is thus the focus of our research group*.

2 MAIN CHALLENGES AND PROPOSED APPROACH

Current software development platforms are tied to a single execution model and hardware platform. Therefore, the end-user developer can usually provide and support their application for a single or very few execution platforms. A few platforms claim that a single software description can be deployed to multiple execution platforms. However, such frameworks are still based on an abstraction level rather low when compared to the needs and the programming skills of the typical application developer. Hence, there is a good chance the developer will not use the full potential of the platform, mainly due to misinformation and lack of deeper knowledge, not to mention the often required tuning to deal with the constraints of the execution platform (power efficiency, restricted memory, etc). Moreover, support for testing, performance analysis, and overall quality evaluation of the resulting application is still very limited, and also requires expertise from the application developer in those topics. For instance, mistakes due to the misuse of available APIs can be hard to be detected. Such mistakes may or may not lead to system failure, performance problems or other non-functional issues, such as power consumption beyond an acceptable range. It is expected that the end-user developer has no freedom (and/or knowledge) in tuning the platform behavior for a given application. Indeed, such a task must be implemented by the platform, in many cases during runtime. Current platform models have little support to such an adjustment.

The keyword for mobile software development is *flexibility*. Flexibility allows the system to cope with the variability in many levels (of the hardware, of the application needs, of the programming languages, etc). It has been long known that one achieves flexibility by *modularization* and *abstraction*. In this aspect, current knowledge of software engineering remains valid. *Reuse* is another important concept, which is also already present in the current development models. These concepts potentially imply higher-quality software, but this must be ensured by methods and tools. What is still missing in the mobile field is the *quality-ensurance-by-methods-and-tools* concept. Much has been said and done about software quality, but this is still not in the mainstream in the mobile field. The second concept that is still missing is *flexibility-in-the-long-run*. How can the platform be flexible enough to realize a change in the application behavior and/or in the hardware behavior?

In this context, we can define five main challenges for the development of a high-level and high-quality mobile software development platform:

1) Definition of appropriate abstractions for each development layer in such a way that: i) the developer of the higher layers can use the provided services without further knowledge; ii) interaction between layers is facilitated to support variability and provide flexibility;

2) Definition of synthesis processes capable of transforming the semantics of the end-user application into cost-effective runtime code while taking into account the non-functional requirements of the application and of the execution platform;

3) Support for application testing and fault-tolerance mechanisms, including mechanisms to deal with uncovered faults in the platform (hardware or software) and misuse of the platform API;

4) Capture some hardware-related issues that have deep impact on the software development process, such as the availability of multicore devices and of distributed computing environments;

5) Support for system evolution in several aspects: execution platform, requirements, programming languages, and so on.

We propose a model-based approach to tackle the mentioned challenges. Current research and results on models are still limited to traditional applications and few tools are available for the support of a complete model-based design. Furthermore, software engineering models are still not capable of capturing behaviors or adapting to the variability in the lower levels. One must find the most suitable models to deal with the specificities of the mobile system: run-time adaptability, variability, constant system evolution, verification requirements, and so on. For instance, a successful model, such as UML, still seems under-utilized due to the lack of supporting automation tools that allow the interaction of the model with the actual runtime code. Our research group proposes a development cycle based on models as a means to achieve both flexibility and quality supported by tools. Models can be verified, can suffer transformations that can be tracked and verified according to different goals, and can serve as a basis for a number of analysis. In our work, we propose the use of models in every abstraction interface, which will require extreme modelling capabilities at all levels, hence the term XModel. Specifically, a research plan towards such a powerful development platform includes the following topics:

- Investigation of distinct abstraction models and the mechanisms for implementing model transformations that can be applied in the different design levels;
- Definition of a formal basis for the different models to enable verification and transformation processes;
- Co-synthesis of fault-tolerant systems, combining hardware and software techniques;
- Model-driven specification and management of platforms;
- On-line and off-line monitoring strategies for verification and optimization;
- Quality metrics applied to embedded software.

3 REQUIRED SKILLS FOR A SOFTWARE ENGINEER

In order to provide adaptive development frameworks capable of abstracting its behavior in favour of expressiveness without losing the possibility of adjustment to the execution platform, a multi-disciplinary approach is required. It should combine traditional computer science knowledge (algorithms, data structures, network, compiling, operating systems, and programming languages) with additional knowledge that is normally not in the mainstream of a computer science course. This additional knowledge would be based on the following ideas:

- Model-based design must be the focus of education: i.e., the software engineer must be capable of defining and using different models as well as relating them through transformations;
- New abstraction models must be explored to capture the semantics of the end-user application. To this end, natural language processing, visual languages, and agent systems theory, for instance, can be considered;
- Verification, Analysis, and Testing concepts are required to support the software synthesis process and evolution;
- Optmization theory can be applied in the synthesis process to deal with the execution constraints and to the run-time adaptation;
- Fault tolerance is also a required knowledge for the platform provider as a means to cope with the escaped faults both in the hardware or in the softwaersoftware platforms;
- Metrics and assessment techniques, which are quite different in the mobile domain w.r.t. the traditonal software development;
- Parallel programming for MPSoC platforms, since this looks like the future of mobile platforms.

REFERENCES

[1] Android Developer site. http://developer.android.com. Accessed on 16 September 2011
[2] RhoMobile site. http://rhomobile.com/. Accessed on 16 September 2011
[3] PhoneGap site. http://www.phonegap.com/. Accessed on 16 September 2011