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.