

MobiCASE 2010
Workshop on Mobile Software Engineering
Position Paper Template

Event: October 28, 2010.
Mail paper to wmse@sv.cmu.edu by July 15, 2010.
No more than 2 pages, 11 point font.

Paper title: Reflections on Mobile Software Engineering

Name: Ray Bareiss

Role/Title: Professor

Organization: Carnegie Mellon SV

Preferred Contact: ray.bareiss@sv.cmu.edu

Please state your point of view on the issues of Mobile Software Engineering. Please specifically address the following 5 questions:

1. Your experience as a software engineer?

I have been involved in software engineering since the late 1970's. I began my career at Tektronix in the (then relatively new) microprocessor support group. I went on to serve as software manager and then director of software engineering for two start-ups. After getting a PhD in computer sciences from UT Austin, I pursued an academic career at Vanderbilt and then Northwestern. I left Northwestern to co-found a software start-up in which I served as executive VP and senior architect. And finally, I joined the faculty of Carnegie Mellon Silicon Valley where I teach in the software engineering and software management programs.

2. Your experience with mobile software development?

My experience with mobile software development has been entirely in the context of courses I teach in software product definition, requirements analysis, and human-computer interaction. In addition, I have advised students on academic and real-world projects involving mobile development.

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

At first glance, the shift from PC-based to smartphone-based application development seems, in many regards, to be strikingly similar to the last major shift from mainframes and mini-computers to PCs in the 1980s, e.g.,

- Lower capability devices than the predominant computing platform (but with rapidly increasing capabilities)
- Simpler, often single function applications
- A different economic model (requiring much cheaper applications to be sold in quantity)
- Shorter development times (driven by different economics)
- Similar but partially incompatible platforms (e.g., different flavors of Android phones, different versions of the iPhone OS) which complicates development and testing, and it may require maintaining variant code bases
- Piecemeal deployment (many end-users making individual installation/upgrade decisions)
- Lack of centralized support.

- Lack of perceived need among developers to use software engineering tools, techniques, and processes given the nature, complexity, and development schedules of the apps they are developing.

That said, even small projects can benefit from adoption of light-weight software engineering processes. Based on a brainstorming session conducted with the Carnegie Mellon Silicon Valley software engineering faculty, to maximize the potential for success, a development team should:

- Engage in user/customer-centered requirements and design processes
- Document requirements, design, and other key project decisions as they are made
- Track progress
- Assess product quality
- Assess client satisfaction
- Maintain effective team processes.

Related to this, faculty believe that all (small) projects should pass “The Joel Test” (<http://www.joelonsoftware.com/articles/fog0000000043.html>). The interesting thing about both our faculty’s brainstormed best practices and those enumerated in the Joel Test is that they are equally applicable to all projects – mobile or not.

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

I believe the main distinguishing features are not in software engineering processes but rather in the functional, technical, and business constraints under which the processes are executed.

- Requirements: Functional requirements are simpler because many current apps are single function; technical requirements may not be achievable at the same level as on the desktop; business requirements are dominated by time to market and development cost.
- Design: Computational, power, and screen real estate limitations constrain design, as does a mobile context of use with potentially varying connectivity.
- Testing: Variability of platforms, connectivity, and usage contexts increase testing complexity.
- Software Process: Many developers seem to have adopted a “release and refine” process.

5. What is the suggested focus and agenda for mobile software engineering research and education?

Given the observations above, I would not advocate changing software engineering education beyond providing a strong introduction to the mobile “ecosystem” and employing mobile projects, emphasizing application of good software engineering processes, throughout the curriculum. Correspondingly, I don’t currently believe that research into software engineering processes need diverge from the path being followed with respect to agile software development; that said, the participatory design aspect of a release and refine development methodology might merit further study. Finally, I believe that research into software engineering tools, particularly development frameworks and testing tools may provide the biggest benefit, at least in the near term.