

Reflections on “Mobile Software Engineering”
Ray Bareiss and Martin Griss
Carnegie Mellon – Silicon Valley

What do we mean when we say “mobile system”? Are we talking about an application that runs on a mobile device, or are we talking about the (usually) much more complex, client-server architecture that supports delivery of the functionality of many of the mobile applications? If the former, relatively little traditional software engineering seems to be taking place; if the latter, development is often conducted by larger organizations and is more aligned with general software industry process. The discussion below focuses primarily on app development rather than on large, client-server system.

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 (moving from centralized deployment to many end-users making individual installation/upgrade decisions)
- Lack of centralized support (at least for non-corporate users but note that Apple’s Genius bar is potentially an interesting partial solution)
- 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, perhaps 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.

More specifically, the faculty posited that best practices for any small project should include:

- A user/customer-centered requirements process and documentation of the resulting requirements via story cards or use cases

- Regular customer meetings to prioritize work
- Documentation of all key project decisions
 - Use of engineering notebooks
- Test-first programming or test-driven development
- Software inspections
- Use of a version control system
- Velocity and/or burndown charts to track progress
- Regular presentation of intermediate versions to customers and potential users
 - “Low-budget” usability testing
- Effective, light-weight meeting management 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. That said, are there any particular areas in which the nature of the mobile environment or the process of developing for mobile devices might raise different issues suggesting the need for additional (or modified) best practices?

Requirements

- Functional requirements may be simpler because of the typical focus on single-purpose apps
- Technical requirements (software quality attributes) may not be achievable at the same levels as in desktop apps (e.g., availability, flexibility, security, reliability, robustness, maintainability, portability, reusability, testability, performance, usability)
- The key business requirements may be time to market and total development cost

Design

- Computational resource and power limitations may constrain design
- Significant constraints on interaction design due to the nature of applications, contexts of use, small screen size, and difficulty of text-based I/O
- A possible trend away from client-server (or at least away from thin clients) towards delivery of substantial functionality on the device itself (to improve availability, reliability, ...) [Are there other changes in design patterns or the emergence of new ones?]

Testing

- Testing is much more complex because of variability of devices within the same family, variability of usage context, and variability of connectivity

Software Process

- Many app developers seem to have adopted a “release and refine” approach to software development in which a new product is rushed to market and then improved over time if it gains traction in the application marketplaces.

Thus Mobile Software Engineering is the selection, refinement and systematic application of light-weight software engineering practices to a highly dynamic domain.