Bringing Cellular Phone Software Industry Practices To The Software Development Classroom

David A. Umphress, James H. Cross II, Larry A. Barowski

Department of Computer Science and Software Engineering
Auburn University, Alabama USA 36849

0. Abstract

University computer science and software engineering curricula are oriented to conventional hardware platforms. This presents an obstacle to teaching students how to develop software for mobile devices. Industry certification practices provide an insight into what is important in teaching software development for a particular class of mobile devices, cellular phones. Adapting such practices to classroom use can aid in bridging the learning gap between traditional and non-traditional platforms.

1. The problem: how to teach cellular phone software development

Cellular phones have become indispensable tools for providing computation resources in a highly mobile package. An estimated 1.5 billion phones are in use today, with the number expected to grow to two billion by 2006[1]. These devices are so pervasive that many developed nations now have cellular communications infrastructures that are more reliable than their legacy land lines. In developed nations, particularly among students and young professionals, cellular phones are increasingly replacing traditional wired phones because they offer mobility, attractive calling rates, and digital services such as messaging and Internet facilities.

Although phone calls are still the primary function of cell phones, these devices are increasingly viewed as mobile computers in their own right. They are equipped with processors that have the computational horsepower to organize address books, manage camera images, run games, and support a wide range of software applications once deemed the unique realm of the Personal Digital Assistant. Over 100,000 jobs in the American software industry involve writing software for mobile devices, with the number expected to triple in the next two years to a significant 11% of the software workforce[2].

The facts are plain: cell phones are a technology to be reckoned with. What are universities doing to equip their computer science and software engineering majors with skills necessary for developing software for such mobile devices? The answer is equally plain: very little[3]. Colleges teach students the principles of software development, but that instruction is oriented mostly to traditional hardware platforms. Developing software for mobile computers requires specific knowledge in working with hardware that is limited in speed, memory, power, and user interfaces. Mobile device software development is still very much a novelty; conventional university curricula do not address these areas to any depth.

Copyright © 2005, American Society for Engineering Education

2. Why is the problem a problem?

Developing software for cell phones – and by implication, mobile devices in general – requires skills and knowledge that differ from "traditional" development for personal computers, workstations, and mainframes. In many regards, the personal computer in its infancy was a harbinger of today's cell phone: those developing software for early personal computers found themselves trading the amenities of working in a relatively cloistered and protected mainframe environment for the stark world of hardware and software at the lowest and most primitive levels of detail. Cell phone developers face a similar situation today: the immaturity of the technology forces them to deal with a myriad of low-level hardware and software issues rivaling early personal computers.

At some level of abstraction, principles of software development for mainframes, personal computers, workstations, or cell phones are practically the same; but, the features offered by the platforms themselves make the difference in writing software a matter of degree. In particular, the nature of cell phones – mobility, small size – requires skills unique to the platform, in addition to fundamental generic software development skills.

Our experience in working with cell phones is in concert with observations by others working in the area of mobile devices[4]. Specifically, writing software for a cell phone requires the developer to balance the following issues:

- How much battery power does the software consume when executing?
- How much memory does the program and its data occupy?
- Can data be stored in persistent memory?
- Is the speed of the processor sufficient for the application?
- Do screen size, color, and resolution match end user needs?
- How is input and output handled?
- How is information synchronized with a desktop system?
- How does the computer communicate with other computers on a network?
- What event models are imposed by the operating system?
- Can software defects affect the integrity of other resident programs or data?
- What are the most effective user interface and computational metaphors?
- How should software applications degrade gracefully?
- What is the appropriate software design technique?
- What development tools and languages are available?
- How portable do applications have to be?
- Are emulators and off-platform development tools available?
- How "real time" does the software have to be?
- How is software provisioned to the device?
- What restrictions are imposed by the device's service provider?

The majority of these issues do not arise when developing software for traditional platforms. Just as no vendor would seriously introduce a software application into the market without first knowing that it works on the target hardware, we should not expect to educate students to develop cell phone software without exposing them to the complexities facing them.

Copyright © 2005, American Society for Engineering Education

Conventional college computer science and software engineering curricula let us down in this regard.

3. Industry software certification as a possible solution

We can look to the cellular phone software industry itself for clues as to how we can structure instruction for our students. Here, the term "quality" has a sharper meaning than it does in the traditional software market. "Quality" connotes software that is robust and reliable; narrowly focused on a single function; does not require user's manuals; and will not reboot the device nor require maintenance upgrades. In short, software, once it is on the phone, must work.

The cell phone industry is attacking the issue of software quality by using a business strategy it has employed with hardware for some time: end-product certification. In this approach, published standards describe what constitutes a suitable software application. Software that has been demonstrated to adhere to the standards is deemed "certified." Only certified software can be marketed and distributed through official cell phone provider channels.

Certification is a simple, yet powerful business model. The quality standards not only describe the character of the software in terms of usability, security, performance, and content, but they also serve to make uniform the look and feel of applications written by different developers. Selling only those software products that have passed certification tests ensures some degree of confidence in the product itself. In addition, it puts economic pressure on the software developer to pay attention to quality, since products that are not certified are unlikely to be sold through cell phone provider official channels and thus will not reach a wide audience.

Qualcomm, with its BREW model, pioneered in 2001 the practice of certification for applications written in the C programming language[5]. Nokia[6], Motorola[7], and Ericsson[8] introduced certification programs for Java 2 Platform Micro Edition (J2ME) in 2002. Sun Microsystems organized a consortium of members of the cell phone J2ME industry in 2003 to create a handset-independent certification process called "Java Verified"[9]. Since then, cell phone software distribution outlets such as Synclast[10] and Cellmania[11] have announced similar, although considerably less stringent, certification measures.

4. Our approach: adapt industry practice for classroom use

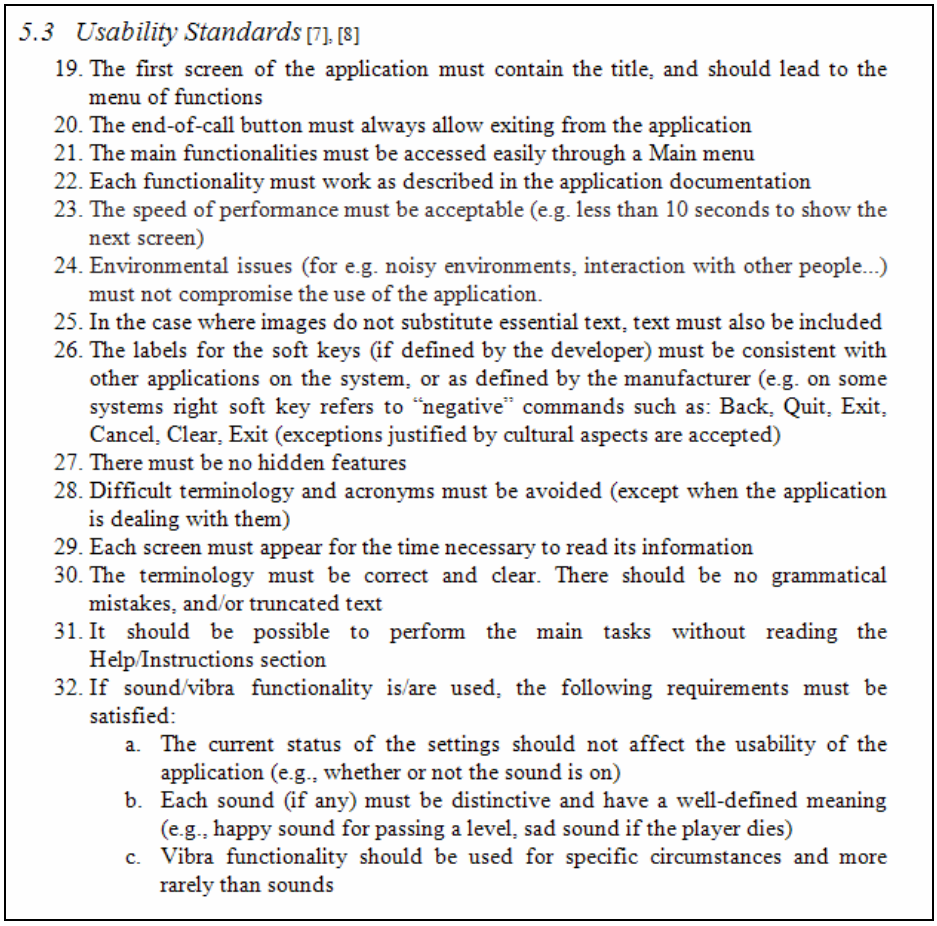
Given the momentum that certification has taken within the cell phone software industry, we felt it could be used in the classroom to both shape course learning objectives and to measure the quality of our students' software. We took a three-prong approach to adapting the certification concepts into the software engineering courses required of our Wireless Engineering majors: standards, tools, and certification. We chose to call our certification process *AU OK*, where *AU* stands for Auburn University and *OK* is a tribute to Nokia's *Nokia OK* program, which inspired the project.

4.1 Standards

We adapted application standards for classroom use as examples of what is expected of industry-quality software. In particular, we drew heavily on the *Java Verified* test criteria[12] and predecessor documents from Nokia's *NOKIA OK* process. References to procedures that were unique to working directly with specific handset were removed, duplication among the standards

Copyright © 2005, American Society for Engineering Education

eliminated, and explanations added to standards that we felt would be unclear to a student audience. Lastly, issues relating to the singularities of developing for software for cellular phones noted above were incorporated into the standard. The resulting lab manual articulates standards on user interface look and feel, human-computer interaction guidelines, security requirements, reliability guidelines, performance requirements, content restrictions, and installation packaging (see Figure 1).



5.3 Usability Standards [7], [8]

19. The first screen of the application must contain the title, and should lead to the menu of functions
20. The end-of-call button must always allow exiting from the application
21. The main functionalities must be accessed easily through a Main menu
22. Each functionality must work as described in the application documentation
23. The speed of performance must be acceptable (e.g. less than 10 seconds to show the next screen)
24. Environmental issues (for e.g. noisy environments, interaction with other people...) must not compromise the use of the application.
25. In the case where images do not substitute essential text, text must also be included
26. The labels for the soft keys (if defined by the developer) must be consistent with other applications on the system, or as defined by the manufacturer (e.g. on some systems right soft key refers to "negative" commands such as: Back, Quit, Exit, Cancel, Clear, Exit (exceptions justified by cultural aspects are accepted)
27. There must be no hidden features
28. Difficult terminology and acronyms must be avoided (except when the application is dealing with them)
29. Each screen must appear for the time necessary to read its information
30. The terminology must be correct and clear. There should be no grammatical mistakes, and/or truncated text
31. It should be possible to perform the main tasks without reading the Help/Instructions section
32. If sound/vibra functionality is/are used, the following requirements must be satisfied:
    a. The current status of the settings should not affect the usability of the application (e.g., whether or not the sound is on)
    b. Each sound (if any) must be distinctive and have a well-defined meaning (e.g., happy sound for passing a level, sad sound if the player dies)
    c. Vibra functionality should be used for specific circumstances and more rarely than sounds

Figure 1. This is a sample page from the *AU OK* certification standard. Other pages include content, system, user interface, development, and deployment standards.

The standards are introduced incrementally across four primary wireless engineering courses, COMP 1210 (Foundations of Computing I), COMP 2210 (Foundations of Computing II), COMP 3710 (Embedded Systems Development), and COMP 4710 (Senior Design Project), as shown in Figure 2. COMP 1210 and 2210 are introductory software development courses using Java. Using a development and emulation environment, we introduce students to the notion that Java is platform independent by illustrating application development with J2ME in COMP 1210. In COMP 2210, students construct a rudimentary application that adheres to the user interface standards. COMP 3710 addresses system-level aspects of wireless devices. Students utilize user interface, performance, and security standards and a development environment to create applications. They first test their software using the emulator environment, and then test their software with actual hardware. COMP 4710 is the point at which students demonstrate their ability to integrate the knowledge they gained across previous coursework. In this course, they

Copyright © 2005, American Society for Engineering Education

develop an application for an actual customer. Their software must adhere to the full complement of standards; they use the wireless development tools; and their software is evaluated internally using the certification process.

		Freshman		Sophomore		Junior		Senior
		Intro to Computing using Java COMP1210	→	**Data Structures using Java** COMP2210	→	**Embedded Systems Design** COMP3710	→	**Capstone Design Project** COMP4710
Wireless industry aspects	Standards							
	UI			√		√		√
	Usability					√		√
	Performance					√		√
	Security					√		√
	Functionality					√		√
	Installation							√
	Tools							
	Editor/compiler	√		√		√		√
	Emulator	√		√		√		√
	Handset					√		√
	Application certification							√

Figure 2. This is the sequence of software engineering courses in the Wireless Engineering software track that take advantage of the certification material, playing an introductory role at the end of the first two courses, and a more prominent role in the last two courses.

4.2 Tools

Although we wish our students to have an exposure to real-world standards, we recognize that we have an obligation to make sure that experience is presented in a pedagogically-sound fashion. We use a particular software development environment, jGRASP, across our computer science, software engineering, and, now, wireless engineering, programs. jGRASP has been demonstrated to be an effective tool for teaching[13] and learning[14], as indicated by its current user base and its inclusion in a number of Java text books. As jGRASP is the intellectual property of Auburn University, we had the ability to tightly integrate other tools into it. Incorporating wireless software libraries, tools, and emulators into jGRASP gave our students the ability to develop cellular phone applications without having to learn another development environment (see Figure 3). jGRASP is available at no cost to the software development community at large (see www.jgrasp.org); the added benefit of this approach is that other educators can take immediate advantage of our work.

4.3 Certification

The final component of our effort to inject industry practices into the classroom is the certification process itself. We took the testing concept of the *Java Verified* program and mimicked it in the classroom. The actual *Java Verified* process has an independent laboratory test the software for conformance to standards. We implemented this process in the small for capstone design projects built in COMP 4710. Senior design teams submit their applications to a team of graduate software engineering students. Working under the mentorship of a faculty member, this group examines the software for conformance to standards and its certifiability. No senior design team is allowed to consider its project complete until it has been certified.

Copyright © 2005, American Society for Engineering Education

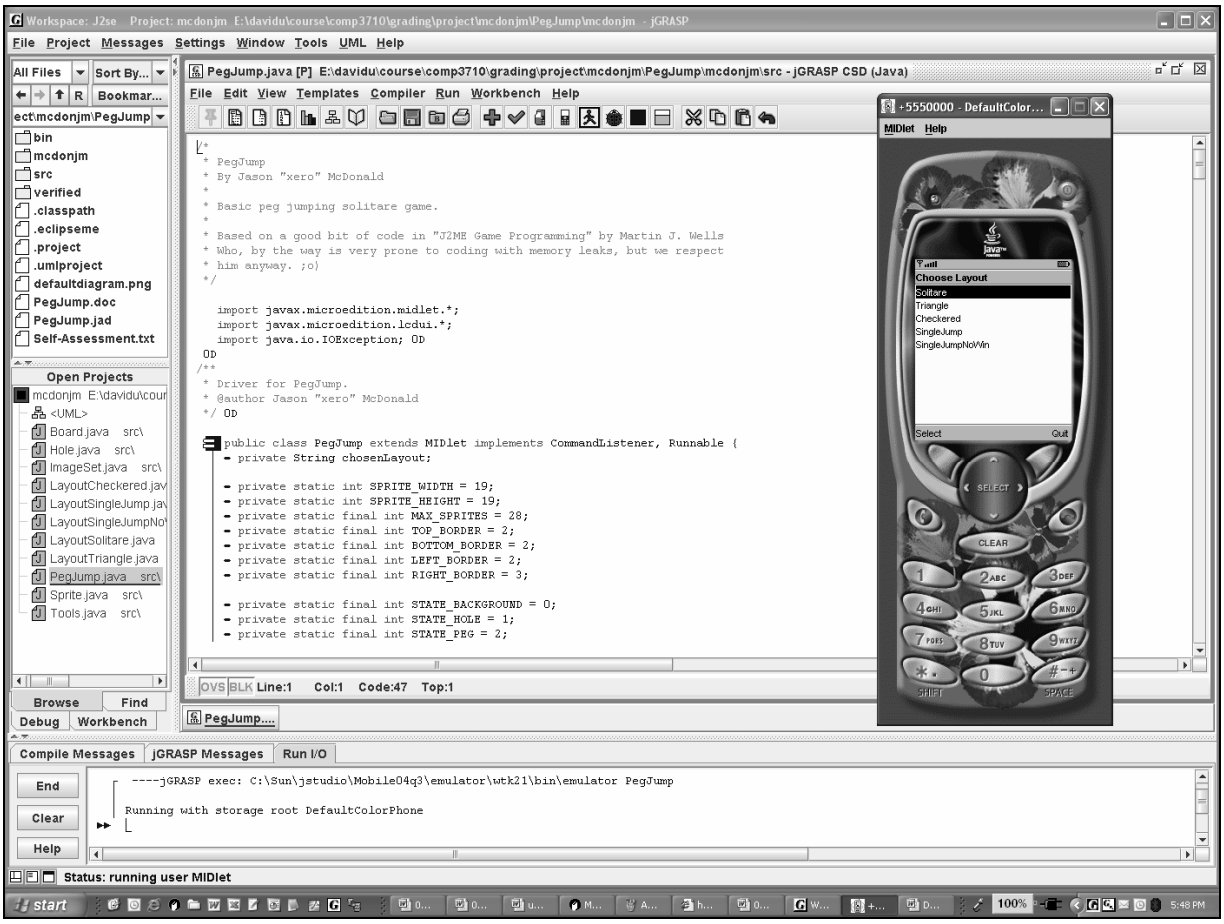


Figure 3. This is a screenshot of jGRASP running a sample J2ME MIDlet, with the default phone emulator is in foreground. Wireless development features of jGRASP allow students to create J2ME projects, compile J2ME code, preverify classes, create JAR and JAD files, and launch the cell phone emulator.

The certification process starts with a student team submitting its J2ME application to the *AU OK* website. (See Figure 4.) The application is then assigned to a test team, who compares the application against externally-verifiable standards, then against design standards, and finally against performance standards. The application receives a score commensurate with the number of standards it meets and in some cases, the degree of standards it meets. Applications scoring below a predetermined quality threshold are returned to the developers for further work. Applications scoring above the threshold are considered "certified."

5. Conclusion

To date we have incorporated the J2ME standards into the primary courses of the wireless engineering degree, jGRASP is being used by the students as their primary development tool, and four J2ME projects have been certified as *AU OK*. The experience in incorporating industry practice into the classroom has been a useful one because it has 1) immersed students in a technology similar to what they are likely to encounter in their first years of work; 2) allowed a non-expert audience to examine wireless industry standards; 3) demonstrated the feasibility of incorporating J2ME development capabilities into a pedagogically-sound software tool; 4) introduced industry testing processes in an academic setting; and 5) provided evidence of the effectiveness of using wireless industry practices in the classroom.

Copyright © 2005, American Society for Engineering Education

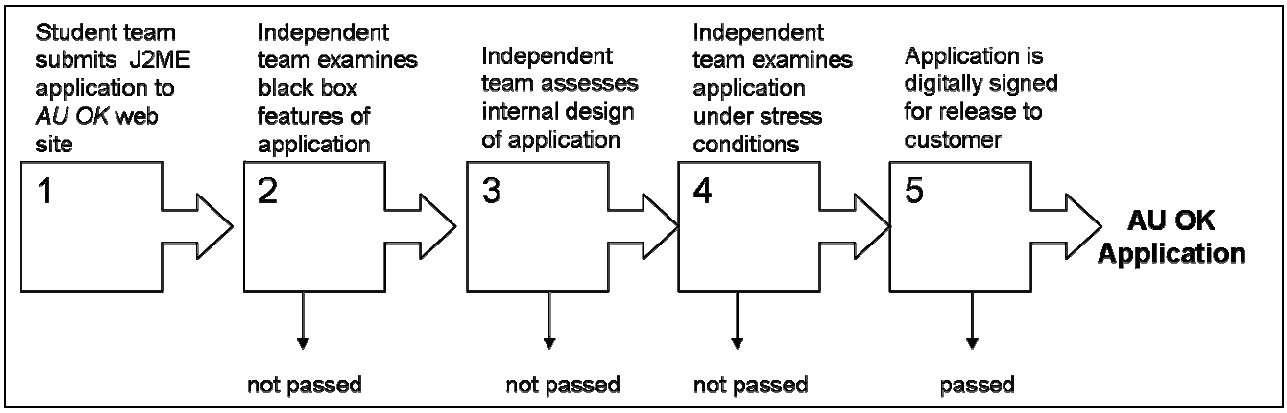


Figure 4. The *AU OK* certification process examines a candidate application for conformance to industry standards that have been adapted to classroom use. Steps 2 and 3 are performed manually; step 4 is carried out with automated test tools.

6. Acknowledgements

This research was supported in part by the National Science Foundation (NSF 0311339).

The authors also wish to thank Nish Meda, John Calagaz, Jhilmil Jain, Udayan Naik, COMP 3710 students, and COMP4710 students for their invaluable student insight into the AU OK J2ME certification process.

7. Bibliographic Information

[1] DAWN. 2004. Number of cell phone users to grow rapidly. http://www.dawn.com/2004/11/06/ebr8.htm.

[2] BLS. 1999. Occupational Employment Statistics for Computer Professionals. U.S. Bureau of Labor and Statistics. http://stats.bls.gov/oes/1999/oes_15co.htm.

[3] Kiely, D. 2001. Wanted: Programmers for Handheld Devices. IEEE Computer 34, 5 (May 2001), 12-14.

[4] Hamburgen, W., D. Wallach, M. Viredaz, L. Brakus, C. Waldspurger, J. Bartlett, T. Mann, K. Farkas. 2001. Itsy: Stretching the Bounds of Mobile Computing. *IEEE Computer 34*, 4 (April 2001), 28-36.

[5] QUALCOMM. 2004. Application and Extension Testing. http://brew.qualcomm.com/brew/en /developer/resources/bd/ext_test.html.

[6] Nokia. 2002. Nokia OK Concept. http://forum.nokia.com/main.

[7] Motorola. 2002. "Motorola Application Certification Program". Version R1.3A. http://qpqa.com.

[8] Sony Ericsson. 2002. Sony Ericsson to deliver tools and services to wireless developers with Metrowerks. 18 Mar 2002 Press Release. http://www.sonyericsson.com.

[9] Sun Microsystems. 2004. The Java Verified Program. http://javaverified.com.

Copyright © 2005, American Society for Engineering Education

10 Synclast. 2004. Java 2 Micro Edition. http://www.synclast.com/app_dist.jsp.

[11] Cellmania. 2004. http://www.cellmania.com.

[12] Sun Microsystems. 2004. Java Verified Unified Test Criteria. http://javaverified.com/docs/Unified_Test_Criteria.pdf.

[13] Cross, J. H., T. Dean Hendrix, and Larry A. Barowski. 2001. "Debugging [in] CS1," Proceedings of 2001 ACM Southeast Conference, Athens, GA, March 16-17, 2001

[14] Hendrix, T. D., J. H. Cross, S. Maghsoodloo, and K. H. Chang. 2002. "Empirically Evaluating Scaleable Software Visualizations: An Experimental Framework," IEEE Transactions on Software Engineering, Vol. 28, No. 5, 463-477.

8. Biographical Information

David A. Umphress is an associate professor of computer science and software engineering at Auburn University. He is the lead investigator of the *AU OK* project.

James H. Cross II is professor of and chair of the Department of Computer Science and Software Engineering Department at Auburn University. He is the director for the jGRASP project.

Larry A. Barowski is a computer science doctoral candidate at Auburn University. He is the lead programmer for jGRASP.

Copyright © 2005, American Society for Engineering Education