Teaching Mobile Computing using Proof-of-Concept and Studio-Based Instruction

Stephen Hughes
Department of Computer Science
University of Northern Iowa
Cedar Falls, IA 50614-0507
stephen.hughes@uni.edu

Abstract

Developing a course on Mobile Computing offers several unique opportunities and challenges. However, it is imperative that courses on emerging technologies, like Mobile Computing, should not only focus on delivering content, but should also use pedagogical approaches that model the skills necessary to take on "the next big thing". In the fall of 2011, the University of Northern Iowa's "Programming for Mobile Devices" course was designed with this philosophy in mind. This paper will elaborate on the design, execution and impact of several experimental approaches used in this course. It will also attempt to identify specific successes and failures of this initial course offering.

1 Motivation

Developing a course on Mobile Computing offers several unique opportunities and challenges. There is a lot of buzz around this topic, but it is by no means an established discipline. A command of the details and specifics of Android or IOS would give students immediate appeal in the job market, while it is still unclear what knowledge provides core competence in this fluid field. Students are excited by the potential and highly motivated to engage in this emerging arena, yet there are few faculty members that they can turn to for an authoritative voice on the subject.

Of course these challenges are not unique to mobile computing; they are endemic to the field of Computer Science. Mobile computing is just the latest in a long line of developments. The same paragraph could have easily been written about web programming ten years ago, and it will likely be written about completely different topic ten years hence. Therefore, it is imperative that courses on emerging technologies should not only focus on delivering content, but should also use pedagogical approaches that model the skills necessary to take on "the next big thing". In the fall of 2011, the University of Northern Iowa's "Programming for Mobile Devices (Android)" course was designed with this philosophy in mind.

Students were informed up front that, unlike a traditional, content-driven course, a significant portion of their efforts would be directed toward exploration, collaboration, failure, discovery and reflection. This environment was supported and evaluated through several distinct course components. Instead of specific outcome-based assignments, students were responsible for engaging in weekly proof-of-concept activities. Students were challenged to develop their own code snippets that isolated some particular concept or skill and demonstrated a reasonable mastery. Students were obligated to reflect on their approach to the course material by maintaining a developer's journal. To encourage collaboration and synergistic learning, all code produced was intended to be treated as class-based open source. Friday class sessions were reserved for public presentation and discussion of student work. These sessions, known as the Friday forum, often took the form of ad-hoc small groups reviewing progress and pitfalls with the weekly concepts. More broadly, many class sessions involved some form of studio time that allowed students to implement, verify, experiment and generally tinker with material being presented.

This paper will elaborate on the design, execution and impact of the approaches outlined above, and attempt to identify specific successes and failures of this initial course offering. The author will share unvarnished student evaluations and will offer insights, lessons learned and modifications to consider while planning future offerings.

2 Course Design

When considering the design for this course, it was necessary to recognize that students were likely to enter with widely varying levels of exposure to the material. Moreover, it was assumed that the novelty and breadth of the topic would spark

student interest differently in a variety of subtopics. Rather than constraining the students to a disciplined schedule of course materials, it was decided that students should be encouraged to explore topics at a depth motivated by their own curiosity. To attempt to tap into a variety of student interests and to maximize student creativity, the flow of the course was driven by the following course components.

2.1 Studio-Based Instruction

This general approach to the class sessions was intended to be consistent with the studio based learning model which "aims to promote learning is a social and collaborative context by having learners construct, iteratively refine, and critically review design artifacts under the guidance of instructors and disciplinary experts." [1]. To promote this classroom culture, several deliberate features were implemented

2.1.1 Class sessions

While instructional time was used to present new information and concepts, a large portion of the class time was set aside class time to actively work on development activities in a collaborative setting. A typical class session began with the instructor introducing a code example or a concept and transitioned into students taking control of the idea and manipulating it individually or in small groups. Students were responsible for ad hoc partnering with class members to compare notes or explore the limits and boundaries of the day's concept. This approach also permitted the instructor to move from the front of the classroom and blend in with student activities as they were engaging the material.

2.1.2 Class-based open source

To encourage an environment of collaboration and exploration, all code that was produced was considered to be open-source for the members of the class. Students were encouraged to review previous submissions as well as collaborate freely with other members of the class. It was expected that students could rely on each other for not only seeing how others approached and solved a problem, but that they could offer explanations of their own code and solutions.

2.1.3 Friday Forums

Friday class sessions were reserved for presentation and discussion of student work; no new material was to be presented. In the initial weeks, these sessions consisted of designed student-led discussions. For example, in the first week, students were tasked with finding, reviewing and explaining the operation of five unique apps. The Friday forum was used to share their findings with their peers. The second week students were charged with finding a press release, news article or some other source that speculates on some aspect of the future of mobile technology. The Friday forum was used to discuss the merits of these speculations. Both of these activities were productive as ice-breakers for the class as well as getting students thinking about the potential for mobile computing in specific ways. After the second week, the Friday

Forums were generally less prescribed and more driven by the issues that the students were directly encountering in their work.

2.1.4 Developer's Journal

The developer's journal was a graded component of the course that was an attempt to encourage reflection and contemplation on process of learning about mobile computing. As explained in the student handout, guidelines for journal entries included:

> 1) Inspirations or Ideas. Find an app that you get really excited about or one that you absolutely despise. Try to pinpoint what it is about the app that got you so worked up. Did you encounter any scenarios where mobile technology would help? Keep your eyes peeled.
> 2) Reactions to the Friday Forum. Even on the Fridays that you present, you will be hearing what other student have to say. What were your reactions to *the ideas* (this is not the place for ad hominem comments)? If things go well, the forum will be a highly simulative experience – you should have lots of questions and comments that emerge.
> 3) Reflections on progress. What concepts did you master? How do you know? How did your beliefs change? What resources worked really well?
> 4) Plans for the future. What issues arose that need further exploration? Do you have specific goals (with timelines)? Can you narrow down a question that you expressed previously?

2.2 Proof-of-Concept Activities

From a practical perspective, the policy of class-based open source code precludes giving assignments with definitive outcomes. Once the solution is derived, everyone has access to "borrow" it. From a pedagogical perspective, the objective is concept mastery, not the production of a specific solution. When challenged to learn something new outside the academic setting, competence is rarely demonstrated to an external authority; one needs to first convince oneself that the topic has been mastered. This dynamic is not captured by traditional, prescribed programming tasks. The alternative was to ask students to engage in "Proof-of-Concept" (POC) activities. As explained to the students:

> Each week you will be responsible for producing an artifact that demonstrates a concept associated with the current discussion topic for that week. This is not intended to be a fully-functioning, polished or complete app – quite the contrary. Instead, you are challenged to develop something that isolates a particular concept or skill and demonstrates that you have a reasonable command of how to manipulate it. You should be prepared to not only share this proof-of-concept with the class, but also discuss why/how it works (or why/how it *doesn't* work).

This approach requires each student to derive their own approach to dealing the relevant concepts. It not only engages their creativity, but also allows students flexibility to explore topics at varied levels according to their interests. Furthermore, it encourages an experimental, incremental approach. Students frequently found an example to work from and tinkered, customized and massaged it to explore the boundaries of the underlying concept.

2.2.1 Structure

The structure of a POC assignment regularly consisted of three parts. First there was a list of topics that should be explored as part of the weeks' activities. These typically were presented as a set of thematically related issue that would likely be consistent with subheadings within a book chapter. For example, an early POC focused on creating user interfaces. Students were directed to explore the issues of activities, views, layouts, styles, menus, toasts, listeners and xml resources. A second component of POC assignments was the expectation that a concept from a previous week would be revisited. Students were directed to either identify a concept that was either "not mastered" or "explored superficially" and expand on their understanding of this topic. To complete this, students were encouraged to consult with some of their peers, do further online research, or ask the instructor to overcome "sticking points". Students were asked to submit their work product along with the final component of each POC: a reflection on what they had/had not accomplished.

2.2.2 Evaluation

In general, the grading of the POC assignments was on the liberal side. The driving principle for the POCs was to keep the students engaged and making progress on learning the concepts associated with the class. Initially, POC assignments were due at the end of the week. After about the fourth week, the deadline was shifted to Monday morning – this allowed students to benefit more from discussions in the Friday Forum.

It should be noted that one of the design principles of this style of assignment was to have *more* work to be done than points awarded. This allowed students to select the concepts that they wished to focus on, as well as the depth with which to explore each concept. The students could earn full credit by either superficially exploring the breadth of the topic, or by extensively drilling into the details of a portion of the topic. This principle also underscored the idea that there is topics need to be revisited; it is not expected to completely master a topic in a single pass through. A typical rubric for POC assignment is listed below.

Your work product should show evidence that you are exploring the concepts for the week. These concepts are assessed as (0) Not evident, (1) Minimal, (2) Extensive, for a maximum of 6 points.

The reflection questions allow you to discuss your progress. Responses to specific questions are evaluated as: (0) Absent, (1) Identified, (2) Discussed, for a maximum of 4 points.

Work Product: 6

2 Activities	2 Listeners	2 Toasts
2 Layouts/Styles	2 xml resources	
2 Menus	2 Previous concepts	

Reflection: 4

2 Outline of Accomplishments
2 Open Questions/Open Issues

2.3 Other Evaluation Components

The Proof-of-Concept assignments served as a mechanism for moving the class through the material while maintaining and measuring student engagement. Given that they were largely driven and directed by student interests, they did not provide a comprehensive measure of student performance. Likewise, there were specific tasks that were required from the entire class, and could not be left to the interests of the students.

2.3.1 Project

To demonstrate general competence as a mobile application developer, students were required to develop a "moderate scale, polished app, or a suite of smaller, related polished apps with a coherent purpose". To give a sense of the scope of the project, students were directed to mirror the quality and functional-level of a 3-star+ apps that one would find on the market.

Students were charged with the task of conceiving of at least three high-level ideas for a project app. A mid-semester Friday Forum was used to pitch the ideas to classmates, brainstorm and refine the ideas, and seek collaborators. Shortly after this activity, students needed to submit a formal proposal that outlined a detailed description of the app, its intended audience and competition, and a list of team members involved in the development of the project. Given that this activity took place relatively early in the semester, students were also cautioned to consider additional Android concepts that would be integral to the success of the project.

As with many traditional concepts, students were required to check-in with progress reports on their project for the remainder of the term.

2.3.2 Tasks

Despite the "open" workload of the proof-of-concept assignments, it was necessary to occasionally ground students with a common, traditional assignment. These were deemed as tasks that it was necessary for everyone in the class to complete by a fixed deadline. These tasks were primarily used to bring the class back together to establish a basis for in-class discussions and investigations. As an example, tasks were relevant at the beginning of the semester for seeding the dialog in the Friday Forums. As mentioned previously, the discussions for the first two weeks stemmed from independent student explorations. Likewise, while discussing Location-Based services, students were tasked with the objective of collecting data on the accuracy of Network vs GPS-based positioning.

2.4 Course Infrastructure

2.4.1 Resources

Given the rapid, ongoing innovation in this field, the value of a course textbook is likely to rapidly depreciate. Rather than assign a required textbook, students were encouraged to make use of online resources as much as possible. In support of topics, the instructor regularly provided links to relevant tutorials and online code samples. Students were also directed to search online developer forums as a first-course of action when hitting a roadblock.

As a general outline for the flow of the course, two trade press books proved to be quire valuable resources for the instructor: Professional Android 2 Application Development [2] and Hello Android: Introducing Google's Mobile Development Platform [3]. These titles were also recommended to students who wished to have a personal reference guide.

2.4.2 Hardware

The Android development environment comes with a platform emulator. While this theoretically eliminates the need for an external Android device, from a practical point of view, a physical device is highly desirable. The emulator suffers generally from poor performance and is cumbersome when it comes to concepts such as Location-based services.

For this course offering, each student was provided with a refurbished, no-contract LG Optimus-T phone. This device or similar entry level equipment can be acquired for approximately $100 per unit. Approximately half of the students enrolled in the course had their own device, and declined the department provided equipment.

3 Student Feedback and Instructor Reflections

The Studio-based classroom and Proof-of-Concept assignments gave this class a very different feel from a traditional computer science course. However, these two components seemed particularly well suited to an emerging topic such as mobile computing. From a student perspective, the class was well received. Overall, over 80% of participating students responded positively with respect to the course effectiveness. Qualitative comments from the students summarizing the course included the following:

> *I enjoyed the format of the class. It was both refreshing and innovative, and placed emphasis on discussion rather than memorize and recite*

> *I enjoyed the material, but I'm not thrilled with the class structure and progress. It all felt rushed and more like a self-study.*

As with any course evaluation, these free-response comments were often contradictory. However, analyzing these comments for this initial offering of the course offers some useful insight for preparing future revisions.

3.1 Friday Forum

> *Loved the open Friday discussions and wish there would have been more full class 'Android' discussions.*

> *I like the Friday Forum – it just didn't result in the sharing of code like I think was expected*

With the exception of the first two planned discussions and the one session related to project brainstorming, the Friday forums were largely unstructured. Students generally utilized this time well to form ad hoc discussions about what they were currently working on. This format was not particularly suited to introverted students. These students seemed willing to engage with the instructor on one-on-one, but struggled with starting conversations with their peers. One of the roles for the instructor during the Friday forum was not only to inject into established conversations, but to attempt to facilitate pairings.

Initially, it was intended that students would regularly be asked to share their progress with the rest of the class, giving quasi-formal code walk-throughs. This would have put more responsibility on the students for demonstrating a command of the material, but students seemed more comfortable with and reasonably productive with small group discussions that emerged.

3.2 Open Source

> *Being a studio course, if all the students code was available for viewing. I think I could have learned a lot by viewing other peoples solutions.*
>
> *One item to note, however, is not to lose sight of the students intellectual property rights for their own apps. Please make clear to future students that using the © symbol will protect those rights.*

One of the failures of this course was that while students were encouraged to share their solutions with other class members, this was not formally supported through the use of a class-wide code repository. The hope was that students would routinely look at another students work and ask, "How'd you do that?" – and that the creator would feel free to share their ideas. I thought that the "on-demand" model would allow the creators to derive a sense of pride from being asked for a solution, while at the same time they would be challenged to justify their coding decisions. In reality, the distribution of code did not fully propagate through the class, and solutions tended to stay within sub-communities within the class. Instituting a class-wide code repository would have somewhat anonymized the code, but would have given a broader set of student-developed examples for students to work from.

Another issue that arose from the open-source policy was related to the student projects. There was a conflict between two students who wanted to create a similar app for their project but did not want to work together as a team. The general idea for the app was discussed during the brainstorming session, yet one student wished to assert intellectual property rights over the concept. On one hand, the basis for the app (a golf assistant) was not unique, but at the same time, it was difficult to permit another student to co-opt the project idea. Resolving this issue took a lot of conversation and ultimately one side was dissatisfied with the solution. Ultimately, this problem was predictable, and in the future, I believe it can be avoided by more careful attention to and management of student expectations with respect to project ideas. I would encourage students to focus on sharable ideas during the class, postponing any commercial ventures until after the conclusion of the course.

3.3 Journal Revision

> *Journal, while a good idea, is difficult in actual practice as I would dedicate 1hr of work but would have difficulty keeping track of what I investigated*

Reflective practice was clearly a pillar to the design of this course. In reflection, however, it was probably overdone. One of the first causalities of the course design was the developer's journal. When the journal was first collected (approximately week 5), it was evident that students were not keeping up with making regular entries. As it turns out, the reflective experience that this component was designed to elicit

was largely redundant with the reflections that were integrated with the weekly proof-of-concept activities.

This component of the course was modified mid-term. Instead of an ongoing journal, students were given the option of submitting a final reflective essay. Students were instructed to offer a macro-view of how programming in the Android environment differs from other programming experiences. In future iterations, of the class, I would retain this component strictly as a final exercise. I believe that this final summative reflection is valuable, but maintaining a dedicated, sustained journal was probably excessive.

3.4 Workload & Expectations

> *The only complaint I have is that this class requires a lot of time outside the class*
>
> *No one had time to really learn anything because they were all too busy trying to complete the insane amount of work for the weekly assignments.*
>
> *Without limitations on how we were supposed to implement our POCs, it was hard to come up with some sort of solid idea of what to make our code do specifically to demonstrate each week's component.*

These observations, and variations thereof, were by far the most common comment on the evaluations. It is somewhat easy for a teacher to treat these comments as a badge of honor: "I really worked them; they must have learned a lot". To some extent, I subscribe to that philosophy. At the same time, I think that the volume of these comments represents a disconnect student-teacher expectations for this learning model.

As described in section 2.2.2, the weekly POCs were intended to offer a set of potential explorations. Students were expected to maintain a sustainable pace that partially addressed the weekly topic list and revisited gaps in subsequent weeks. Despite regular conversations about this in class and out, students had a hard time reconciling these expectations with their traditional classes. This disconnect manifest itself in two ways. First, there were ambitious students who erred on the side of overachieving. These students faced the potential of burning out and ultimately missed submissions near the middle to end of the semester. The inverse problem arose with students who routinely seemed uncomfortable submitting "incomplete" work. These students missed submissions early in the semester, and had a harder time getting into the flow of the course.

I believe that this problem requires active and regular intervention from the instructor to better communicate the expectations associated with the POC assignments. In addition to the open discussions on the topic, emphatic feedback is essential on the early POC assignments. Moreover, as students experience this classroom model in a variety of courses, they will gain a better understanding of expectations.

References

[1] N. H. Narayanan, C. Hundhausen, D. Hendrix and M. Crosby, "Transforming the CS Classroom with Studio-Based Learning," in *SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education*, Raleigh, 2012.

[2] R. Meier, Professional Android 2 Application Development, Indianapolis: Wiley Publishing, 2010.

[3] E. Burnette, Hello Android: Introducing Google's Mobile Development Platform, Pragmatic Bookshelf, 2010.