Education & Training

Smartphone Computing in the Classroom

Jules White and Hamilton Turner

Editor's Intro:

Smartphone computing platforms are increasingly used for instruction because such devices are becoming as common as traditional desktop computers and they can excite students about computing and networking. This column describes a network application design course at Virginia Tech that uses smartphones as computing platforms. It seeks to provide in-depth descriptions of important and innovative work in education and training in pervasive computing. I welcome your suggestions and comments for future columns. ***—Scott Midkiff***

Smartphones, such as the iPhone and Google Android, have become extremely popular and constitute an ever-increasing share of computing platforms. For example, in the third quarter of 2010, 88.3 million PCs were sold worldwide. In that same quarter, 80 million smartphones were purchased. Roughly 20 million Android devices and 14.1 million iOS phones were sold. Moreover, whereas there was only a 7.6% growth in PC sales from the previous quarter, smartphones platform sales experienced approximately 30% growth.

The processing capabilities of these smartphone devices have generated significant interest in—and development of—third-party applications. Most notably, the Apple application store contains more than 100,000 applications and has had more than 1 billion downloads to iPhones. It reached the 500 million download mark in January 2009 and the billion download mark by April 2009. These devices possess impressive hardware capabilities, as well as powerful software distribution, upgrade, and maintenance platforms supported by these application stores. Already, many of these applications possess cyber-physical[1] qualities and supporting cloud services.[2] For example, Google's Latitude service uses a client-side application to capture location information from GPS sensors and intelligently aggregates information through a centralized cloud service that provides features, such as alerting you when you are near other friends that use Latitude.

Given the increasing use of and excitement surrounding smartphone computing platforms, a key question is how they can be leveraged to enhance computing education. Here, we describe the ECE 4564 Network Application Design course at Virginia Tech, which we structured to use a combination of smartphone platforms, cloud computing, and open-ended assignments. We present the challenges that we faced when designing the course, our solutions to those challenges, and data measuring independent learning outside of the classroom. Our experience teaching ECE 4564 and the results of our analyses of student learning show that smartphones have significant potential for improving self-directed student learning outside the classroom**. //Author: I/'m not sure how the highlighted words fit into the sentence. Do you mean improving self-directed learning outside the classroom as well as learning core topics …?//**. – Editor: please just cut the words from the sentence.

ECE 4564 Overview

Virginia Tech's ECE 4564 Network Application Design course focuses on teaching students the concepts involved in designing and implementing high-performance and extensible network applications, such as webservers. The course introduces students to network application design topics, such as:

- TCP/UDP/IP and quality of service (QoS),
- Designing application-level protocols,
- Application programming interfaces (APIs),
- Socket programming,
- Concurrency and synchronization with processes and threads,
- Software design patterns for concurrency and networking,
- Testing and debugging networked applications, and
- Client/server design.

A key aspect of the class is understanding how to develop concurrent network applications using software design patterns, such as leader-followers, half-sync/half-async, active object, and futures.[3] Students explore these concepts in the course by designing and implementing a series of client-server applications.

In previous years, the course was taught in the context of desktop clients and server-based networked applications using technologies such as Microsoft's .NET. Our goal in redesigning the course was to cover the same concepts but in a mobile computing context with smartphone clients and cloud-based servers. We hoped that smartphone-based assignments and code examples would allow students to more easily relate the coursework to their external interests.

Adopting a Smartphone Computing Platform

The redesigned course was built around in-class lectures that introduced patterns, concurrency, protocols, and other topics in the context of client-server smartphone applications. To incorporate smartphone topics, we made several key changes.

First, all assignments were open-ended and could be completed by building any mobile application of the student's choosing. Code examples and assignments used the Google Android smartphone platform and the Google App Engine cloud computing platform. In addition, example and student code was open sourced with the Apache 2 License. Finally, we expressly encouraged students to use third-party open source libraries.

Mobile and Server-Side Platform Selection

Choosing mobile application and server-side platforms was an important aspect of the course design. Initially, we chose two candidate platforms for the class: Google's Android platform and Apple's iOS. Both platforms have significant developer communities and resources to support students. The development tools for iOS, particularly the user interface builder and emulator, were more polished but required students to have access to OS X computing resources. Moreover, the requirement for students to learn Objective-C, was a daunting challenge. In the end, we selected Android for two primary reasons:

- Students did not need to purchase any licenses to develop on real devices, and
- The developer tools are cross-platform, allowing students to develop on OS X, Windows, or Linux.

Most of the student development was conducted on the Android Emulator, which is a software emulation of an Android device. In addition to providing accurate software behavior, the emulator allows students to adjust many of the emulated device's networking and physical properties. For example, students can dynamically change the simulated network connection to mimic WiFi, cellular Enhanced Data rates for GSM Evolution (Edge), or 3G latency and speed. The physical aspects of the phone, such as the GPS coordinates reported by its location API, can be scripted to simulate movement through a specific geographic region. Third-party libraries are also available to send mock accelerometer data. The instructors had a total of 10 Motorola Droid smartphones that students could check out. In practice, we found that a large number of students already had Android-based phones and that 10 Droids were more than enough to support a class of 43 students.

Although we were initially concerned because students had little exposure to Java, which is the language used to develop Android apps, the combination of extensive configuration wizards, community resources, and open source code allowed students to easily gain the implementation skills needed to complete projects. In fact, the instructors often found themselves researching an advanced framework, outside the scope of the class, that a student was using. For example, several students developed applications that used facial recognition and text to speech. Implementation details and a new programming language didn't appear to detract from student learning.

Assignments with Student-Selected Contexts

A characteristic of many of our best students was a desire to independently learn advanced topics beyond those covered in the course. These students were often the most engaged in class, asking and answering

questions to further their understanding of a topic. Very often we've found that these self-motivated students are working diligently on their own to apply the concepts to personal projects.

Open-ended assignments. A question that we asked ourselves was could we tap into intense student interest in other subject matters to produce more students that exhibited the independent drive of our best students? In particular, we wanted to increase the energy that students spent on the course assignments and their interest in independently learning advanced topics beyond the core scope of the class. To achieve this goal, we devised assignments with requirements that could be demonstrated in the student's choice of context. Often, we've heard students complain of the endless hours spent on assignments that they feel have no value other than to help them make an acceptable grade. Our approach was to elicit the opposite reaction—to let the students choose a context they cared about. As an example, the first class assignment is shown in Figure 1.

Assignment 1

Directions: Create an app that uses HTTP communication. The requirements for the app are as follows:

1. Your app must use an HTTP GET request to obtain data.
2. Your app must use an HTTP POST request to send data to a server.
3. The app must not block the GUI thread while performing HTTP communication.
4. The app must use the data obtained from the server to populate a GUI.
5. The GUI cannot consist solely of one text box or label.
6. If you use a third-party library, such as the Android Facebook SDK, any HTTP communication that it performs **//must?//** satisfy requirements #1 & #2.
7. Prepare a 5–7 minute presentation describing what the app does, how it works, and its design.

Figure 1. A sample class assignment asking students to develop an application in a context of their choosing ***//Author: OK? Please revise as necessary.//****. //Editor: Looks good*

Because the class was taught in the context of mobile network applications, it was easier for students to connect the requirements to their other interests. A typical statement from a student about why he or she chose a particular project to fulfill the requirements was, "there just aren't any apps out there to do X."As we discuss later, the students satisfied the assignment requirements in a diverse set of contexts, ranging from assistive technologies for friends with Parkinson's disease to cyber-physical games, such as Human Pong, that blended real-world and cyber aspects into the game controls.

Open source mobile applications. To avoid intellectual property issues, we used open source code examples and student applications. We licensed code under the Apache 2 License and made it freely available through a Subversion version control system hosted on Google Code. Although we used open source licenses, we retained copyright of all code that was produced.

A ramification of the open source model was that students had access to each others' source code. With a traditional assignment structure, where the bounds of assignments are strictly defined, allowing students to see each others' work would create the possibility that students would copy each other without grasping course material. Because we chose a structure that required students to choose their own context (design any mobile app they desired) to complete assignments, students could not simply copy each others' work. No two apps designed by students were the same and there was no way of directly copying code without significant modifications, which students could only achieve by applying course concepts. In some cases, students specifically designed modularized libraries for use by other students, such as a simplified speech recognition framework for Android.

Evaluation of Independent Learning

A key goal of the course design was to leverage a smartphone-based curriculum to stimulate learning outside the curriculum. Often, course evaluations and questionnaires are used to gauge student interest in

course material and perceptions of class organization. However, the responses returned, regardless of how positive, might not actually indicate student interest that translates into improved comprehension of course material or future motivation to further explore the course material outside of class. Demonstrating a link between student interest and students' ability to independently assimilate information and apply it to real-world computing problems isn't easy.

Independent learning. We looked at the source code for 122 Android and 1 Windows Phone7 app produced in the class. From this analysis of the open source code produced by the students, we built a list of the concepts that were used and taught in class as well as the concepts that students independently learned and applied in their projects (see Figure 2).

*Figure 2.Core concepts taught in class and the concepts independently acquired by students, as evidenced in their applications' source code //**Author: Please feel free to revise.**//. Editor: Looks good*

The applications demonstrated a total of roughly 67 independently acquired concepts. Approximately an equal number of concepts were directly taught in class and applied to the projects. Certainly, there are multiple ways of classifying and counting the independently acquired topics. For example, students used the HTTP cookie infrastructure in their projects, which is an extension of the HTTP client infrastructure taught in class. Regardless of the methodology used to precisely count independently acquired topics, roughly an equal amount of material was learned inside and outside of class. We believe and intend to more rigorously prove through analysis of future classes that this large number of externally acquired topics was directly related to significant student interest in smartphones and the ease of applying open-ended smartphone-based assignments to the students' chosen context.

Students used a wide array of contexts for their applications. The following is a rough taxonomy of the topics that the student applications //**addressed**//: //**Okay to alphabetize?**// **Editor: Sure!**

- Accessible technologies
- Agile development
- Audio/video streaming
- Campus life
- Chat
- Cyberphysical gaming
- Emergency response
- Finance
- Food
- Geolocation
- Multiplayer gaming
- Music
- Productivity
- Reference
- Shopping/commerce
- Social applications
- Specialty art
- Sports
- Transit
- Translation
- TV
- Weather

A large percentage of the apps, when compared to the apps available through the Android Market, appear to be unique.

Future student research and independent studies. One positive outcome of the class was that many students expressed interest in continuing their studies on network smartphone applications. Of the 43

students in the class, nine (roughly 21 percent) planned to pursue independent studies or undergraduate research related to networked smartphone applications. Another interesting result was that three teams of students contacted venture capital firms to inquire about funding ideas. Two of those teams are forming startups and one is currently pursuing Small Business Innovation Research (SBIR) funding from the Department of Homeland Security.

Our experience teaching network application design in the context of smartphones and cloud computing taught us several valuable lessons. First, although we were initially concerned that the implementation details and requirement to learn a new programming language could detract from learning, we did not observe any problems. Moreover, we found that students' exceptional interest in smartphones and their excitement for building mobile apps that solved problems they cared about motivated them to independently learn a number of advanced topics outside the scope of the class. Another key lesson was that by letting students build apps of their choosing and only dictating minimum requirements, students could openly share code with each other and collaboratively brainstorm solutions to problems without directly copying code.

The lectures, example code, exercises, and open source applications are available from the course website at http://code.google.com/p/vtnetapps.

References

1. J. White et al., "R&D Challenges and Solutions for Mobile Cyber-Physical Applications and Supporting Internet Services," *J. Internet Services and Applications*, vol. 1, no. 1, May 2010, pp. 45–56.
2. H. Erdogmus, "Cloud Computing: Does Nirvana Hide Behind the Nebula?" *IEEE Software*, vol. 26, no. 2, 2009, pp. 4–6.
3. D.C. Schmidt et al., *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, vol. 2, John Wiley & Sons, 2000.

Jules White *is is an Assistant Professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. He received his BA in Computer Science from Brown University, his MS and PhD from Vanderbilt University. His research focuses on developing mobile cyber-pysical systems using smartphones, designing power-efficient communications middleware for smartphones, and applying search-based optimization techniques to the configuration of distributed, real-time and embedded systems. Contact him at julesw@vt.edu.*

Hamilton Turner *is a PhD candidate in the Department of Electrical and Computer Engineering at Virginia Tech. His research interests include* ***smartphone-based sensor networks, multicore cache optimization techniques, and algorithmic deployment optimization of distributed systems****. Turner has a BA in Computer Science from Vanderbilt University. Contact him at hturner0@vt.edu.*