Contextual Android Education

James Reed David S. Janzen

Abstract

Advances in mobile phone hardware and development platforms have drastically increased the demand, interest, and potential of mobile applications. We report on development of a new special topics software engineering course that combines the appeal of Android application development with software engineering topics and entrepreneurial thinking. The primary contribution of this project and the focus of this paper is a series of detailed educational laboratory exercises that are designed to supplement the Android documentation by providing contextual examples, activities, and tutorials. The labs were contributed to the Google Code University under the Creative Commons license, resulting in over 30,000 visits and nearly 100,000 pageviews in its first three months.

1. Introduction

With advances in mobile phone technology, the demand for both free and paid mobile applications has risen dramatically. As a result, mobile phones have become a popular new medium for application development. There are many high powered platforms to choose from, including Blackberry, iPhone, and Android. Most of these platforms offer an open marketplace for the sale and distribution of independently developed mobile applications which makes it relatively easy for independent developers to get their products into the hands of users. Awareness of this new market has not escaped the interest of computer science (CS) and software engineering (SE) students who are often consumers of the applications themselves. With the demand for mobile applications, low barrier to entry into the market, sophistication of mobile development platforms [1], and general interest from the CS/SE student population, mobile application development is an excellent skill for computing students to learn. Most of these platforms offer a wide array of documentation for learning how to develop for their platforms; however, the body of knowledge for contextual examples and tutorials is drastically smaller in comparison. This can be troublesome for some college students attempting to break onto the mobile development scene on their own.

Since January 2010, California Polytechnic State University has offered a mobile development class that teaches students how to write applications for phones running on the Android platform. This class aims to take advantage of students' current interest in mobile applications. In addition to providing the students with an in-depth knowledge in mobile development practices and the skills necessary to be successful Android application developers, the class incorporates aspects of Test Driven Development, Agile Processes, Design Patterns, application profiling, engineering for performance, and entrepreneurial thinking into the curriculum.

We report on a series of detailed educational laboratory exercises developed for this course. These labs were designed to supplement the Android documentation by providing contextual examples, activities, and tutorials. They were designed and used in coordination with in-class lectures and a ten-week course project. The course project imposed the use of software engineering best practices and a software engineering process on small teams. While the practices and process encompassed many of the core SE learning objectives of the course, this paper focues on the laboratory exercises because we believe they are the most novel contributions of the project and will provide the primary value to CS/SE educators. The remainder of this paper will discuss the content of the six labs, with a brief conclusion.

2. Lab Details

The following sections overview the six labs. Some user interface examples are shown. Complete lab specifications are available at http://sites.google.com/site/androidappcourse/.

2.1. Lab 1

The goal of the first lab was to teach students the fundamentals of developing Android applications, from project creation to installation on a physical device. More specifically, it was intended that students would learn how to use the basic development tools to support the application development process, as well as the major components of an Android application itself. The lab accomplishes this by having the students setup their own development environment from scratch, develop a basic "Hello World!" application, run the application on the Android Emulator, and deploy it to a physical device.

2.2. Lab 2

The second lab teaches students how to work with Android's user interface (UI) library by creating a "Joke List" application that allows a user to view and edit a list of jokes. In this lab, students learn the following skills:

- **Executing Tests:** Automated JUnit tests are distributed with the labs to help ensure that students are implementing the labs correctly and to provide students with a source of instant feedback. Students are taught how to execute a single test class from Eclipse and how to make use of the test results to identify errors.
- **Referencing Resource Data:** Lab 2 demonstrates how to work with Android's system for externalizing resources like images, strings, and UI layouts.
- **Declaring Dynamic Layouts:** Students learn to dynamically manipulate a UI at runtime while getting practice working with different View classes and their interfaces in a familiar programmatic paradigm. In Lab 3, students are then taught how to declare and reference UIs as externalized layout resources.
- **Handling UI Events:** Students learn about event-driven programming by implementing event listeners and registering them with their associated View objects. They create and register listeners for touch and key events that allow users to add jokes.

Figure 1. "Advanced Joke List" Application Screen Shots

2.3. Lab 3

Lab 3 continues Lab 2 by creating a more polished interface and adding functionality to the "Joke List" app. Students first learn how to turn their UIs into externalized resources by declaring them statically in XML layout files. Then students learn how to extend the Android UI library by implementing their own custom View class that is used to display jokes in both an expanded and collapsed mode as shown in Figure 1. Lastly, students learn how to create menus to filter jokes and HTTP Connections to exchange jokes with a server.

Students incorporate their custom joke View class into the "Joke List" application by using AdapterViews. AdapterViews are View objects whose child Views are determined by an Adapter, which binds to a data source[1]. The Adapter creates a View object for each piece of data. In the "Joke List" application, the AdapterView is the scrollable list of JokeView objects. These are provided to it by a custom JokeAdapter which creates JokeView objects for the array of Joke objects to which it is bound.

2.4. Lab 4

Lab 4 builds on Lab 3 by adding state persistence to the "Joke List" app. In previous labs, all joke data and internal state is lost when the application is closed. For this implementation, students learn to persist joke data by using an SQLite Database, and preserve internal state by using a combination of mechanisms supplied by the Android platform.

- **Persisting Application Data:** The Android platform offers each application the ability to create its own private SQLite database. Students learn how to persist data through the use of an SQLite database adapter that has been implemented for them. This implementation comes in the form of a closed source jar file. Once the students understand how to use the database adapter, they learn how to implement their own.

- **Maintaining Internal State:** Students learn to maintain internal application state by monitoring the lifecycle of their application and saving and restoring state at the correct times. The students use Instance State to manually save and restore the text in text fields. The students also use Shared Preferences to allow the application to remember which joke filtering option was chosen by the user.

[1] http://developer.android.com/guide/topics/ui/menus.html

2.5. Lab 5

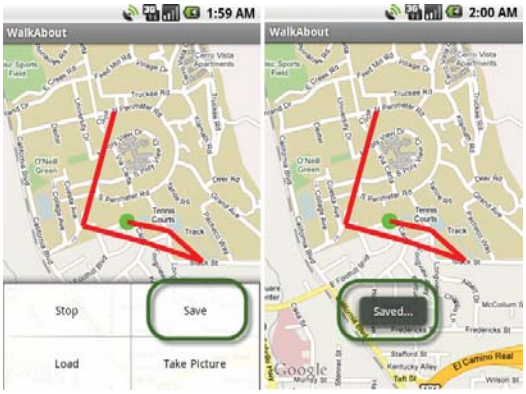

Figure 2. "WalkAbout" Application Screen Shots

For this lab, students develop a new GPS recording application called WalkAbout. The purpose of the application is to allow users to record their GPS location information as they travel. While the application records the user's GPS data, it displays it back to the user in the form of a path drawn on top of a Google Map. While recording data, the user can launch a Camera Activity that will capture and store pictures on an SD-Card. When finished recording, the application gives the user the option of storing the current GPS data as a private application file to be loaded and displayed at a later time.

- **Using Google Maps & Overlays:** Students learn how to display maps by using the MapView and MapActivity classes in the "Walk About" application. These classes encapsulate all the viewing and gesture logic necessary for handling panning, zooming, and touching objects on a map. Students then learn how to implement their own overlay to display a path. The path overlay the students create draws a red path on a map for a given list of latitude and longitude points.
- **Using GPS:** Students interact with location-based services by adding functionality to monitor the GPS hardware in the "Walk About" application. They record changes in location as the user's path which is drawn onto the map.
- **Using a Camera:** Students learn how to use camera-related API's by implementing a camera Activity that allows users to take pictures from the "Walk About" application. This activity displays a full screen preview of what is seen through the camera.
- **Working With Files:** Students learn how to use internal file storage by adding functionality to the "Walk About" application to save and load a user's path as an internal file. Then they learn how to use external storage by saving a picture taken from the camera Activity as an external file on an SD-Card.

2.6. Lab 6

For this lab, students develop a new application named AppRater that suggests other applications for users to download and rate. The application makes use of additional Android application components, including ContentProviders, Services, and BroadcastReceivers.

ContentProviders allow content to be shared and edited by other application components much like a database adapter. Students learn to interact with a provided ContentProvider that serves their "App Rater" Activity with suggested applications to display and rate. This comes in the form of a closed source jar file. Once the students understand how to use a ContentProvider they learn how to implement their own.

Services are used to execute code which needs to run regularly, but does not need a user interface. They can be started from an Activity or awoken by System Notifications. Students learn how to launch a Service that has been implemented for them which refreshes the "App Rater" application with new applications to rate.

Intents are used by the Android platform as a System-level message passing system. They can be used to start application components, or to pass messages between components. Students learn how to do both by implementing a BroadcastReceiver to listen for Intents that they broadcasted from their "App Rater" download Service.

3. Conclusions

We have presented labs that teach students about a broad range of Android topics with enough depth to enable them to develop Android applications on their own. Based on survey responses, students think the labs are interesting, clearly written, and challenging. Students also feel that the labs accomplish their stated learning objectives and prepared them to work independently. In fact, all students were required to deploy their course project apps publicly to the Android Market. Many of these continue to exist in the Market, and at least one has prompted the formation of a startup company. Android provided a compelling context for discussing software engineering topics such as design patterns and efficiency, and applying practices such as test-driven development and code reviews.

References

[1] Gronli, Tor-Morten and Hansen, Jarle and Ghinea, Gheorghita, *Android vs Windows Mobile vs Java ME: a comparative study of mobile development environments*, Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments, 45:1–45:8, New York, NY, USA, ACM, 2010