Presenting Android Development in the CS Curriculum

Mao Zheng
Department of Computer Science
University of Wisconsin-La Crosse
La Crosse WI, 54601
mzheng@uwlax.edu

Hao Fan
International School of Software
Wuhan University
Wuhan, Hubei, 430079
hfan@whu.edu.cn

Abstract

Mobile application development is a hot topic that has great appeal to computer science students. Many institutions are considering offering a course on mobile application development. Some institutions already have. There are two major platforms in the mobile device community: the iOS and the Android platform. The authors are interested in presenting Android development in the CS curriculum for a number of reasons: Android provides a rich platform with a variety of concepts, techniques, and resources which can be combined to produce useful and marketable applications. In addition to its openness, all the tools in the Android development are free and no special hardware is required. The programming language in Android development is java, which is the current programming language of choice for courses CS1 and CS2, offered by the Department of Computer Science at the University of Wisconsin-LaCrosse. However, we also aim to use Android technology to illustrate software design principles. The purpose is to introduce/emphasize software design and Object-Oriented concepts using a real platform case, and to deepen the students' programming skills in advanced Java development. For example, the Android architecture is clearly an illustration of the design principle "separate of the concerns". Each application starts from an "Activity". An activity represents a single screen with a user interface. The user interface is defined in the activity's layout xml file. Hence, what the screen looks like and its behavior are naturally separated into different files in the Android architecture. This separation allows the interface to remain unchanged when only the behavior needs to be changed. The opposite is also true.

The authors hope the discussion in this paper will explore the idea of using mobile learning to teach computer science principles. The idea has two purposes: one is for upper division students to examine the theoretical principles in computer science and how they are related with the use of real world applications. The second is for intro-level students to better understand the Android's architecture and to practice advanced object-oriented programming.

1 Introduction

Mobile application development is a hot topic that has great appeal to computer science students. Many institutions are considering offering a course on mobile application development. Some institutions already have. There are two major platforms in the mobile device community: the iOS and the Android platform. The authors are interested in presenting Android development in the CS curriculum for a number if reasons: Android provides a rich platform with a variety of concepts, techniques, and resources which can be combined to produce useful and marketable applications. In addition to its openness, all the tools in the Android development are free and no special hardware is required. The programming language in Android development is java, which is the current programming language of choice in CS1 and CS2 in the Department of Computer Science at the University of Wisconsin-LaCrosse. However we also aim to use Android technology to illustrate software design principles. The purpose is to introduce/emphasize software design and Object-Oriented concepts in a real platform example, and to deepen the students' programming skills in advanced Java development. For example, the Android architecture is clearly an illustration of the design principle "separate of the concerns". Each application starts from an "Activity". An activity represents a single screen with a user interface. The user interface is defined in the activity's layout xml file. Hence what the screen looks like and its behavior is naturally separated into different files in the Android architecture. This separation allows the interface to remain unchanged when only the behavior needs to be changed. The opposite is also true.

The authors hope the discussion in this paper will explore the idea of using mobile learning to teach computer science principles. The idea has two purposes: one is for upper division students to examine the theoretical principles in computer science and how they are related with the use of real world applications. The second is for intro-level students to better understand the Android's architecture and to practice advanced object-oriented programming.

2 Android Technology

The Android operating system is developed by the Open Handset Alliance led by Google. It includes a large set of features for supporting mobile applications. Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an *Application Framework* which includes Java-compatible libraries based on Apache Harmony [2]. Android uses the Dalvik virtual machine with just-in-time compilation to run compiled Java code. The Android development environment includes a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE. The programming language is Java.

The emulator available in the Android SDK is a tool that allows developers to easily test applications without having to install it to a real device. With the proper configuration for

an emulator, it is also possible to test situations which are hard to reproduce on a physical device.

2.1 Android's Main Components

Application components are the essential building blocks of an Android application. There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed [2].

2.1.1 Activities

An *activity* represents a single screen with a user interface. A multi-screen application will consist of a number of activities that work together to form a cohesive user experience. However, each activity is independent of others. An application can start any one of these activities.

2.1.2 Services

A *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. Example: a service might play music in the background, while the user is in a different application. An activity can start the service and let it run or bind to it in order to interact with it.

2.1.3 Content Providers

A *content provider* manages a shared set of application data. Through the content provider, other applications can query or even modify the data (if the content provider allows it).

2.1.4 Broadcast Receivers

A *broadcast receiver* is a component that responds to system-wide broadcast announcements.

Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message called an *intent*. Intents bind individual components to each other at runtime.

2.2 Android Application Structure

Figure 1 below shows an Android project typical structure in Eclipse. The java files are located under the ***scr*** folder. The interface is defined in the xml file under res/layout folder. The AndroidMainFest.xml file will define the start activity of the application.

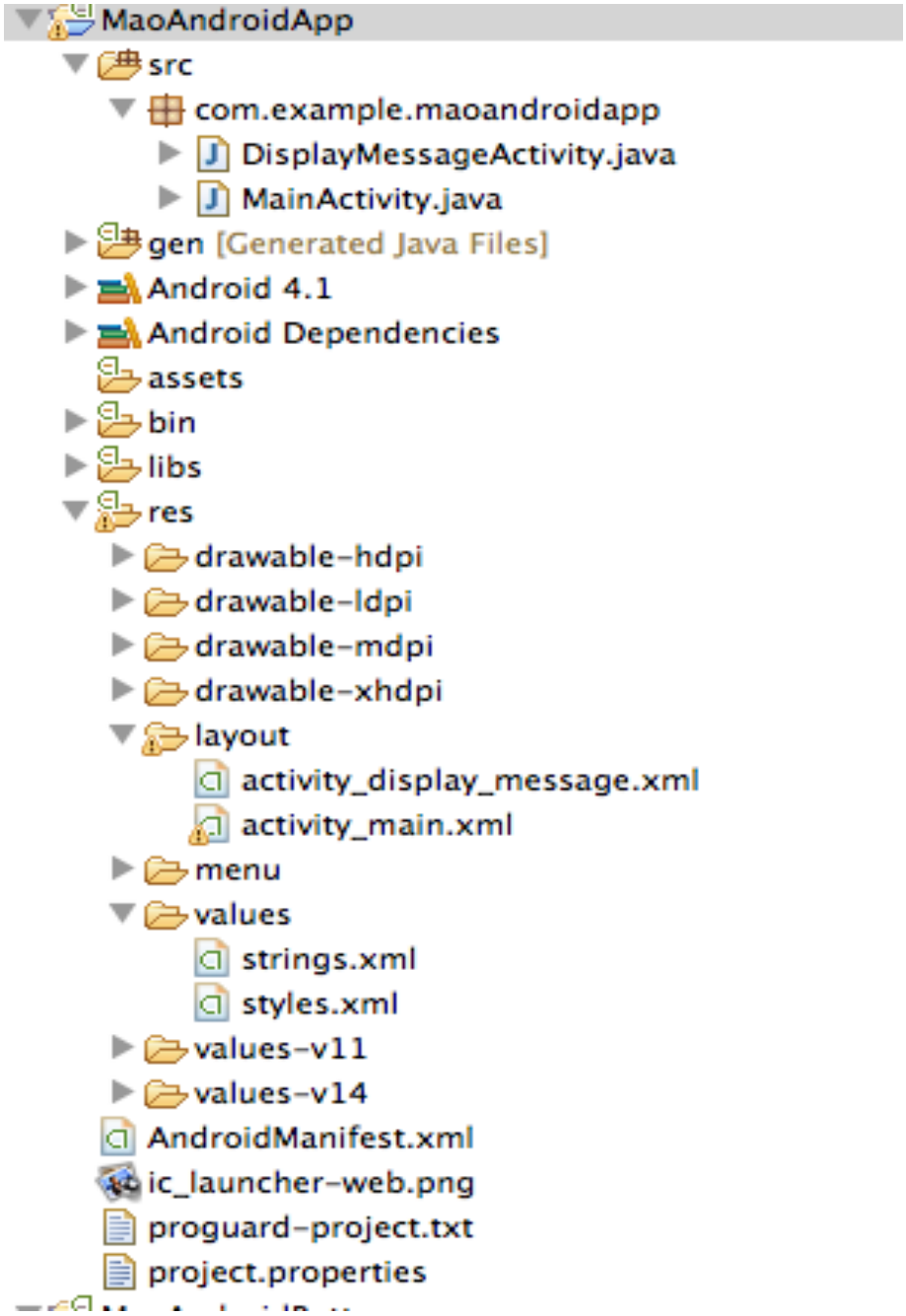

Figure 1 Android Application Structure

3 Separate of the Concerns

Separate of the concerns is a design principle to control complexity in the software development. The idea is that a software system must be decomposed into parts that have minimal overlap in functionality. It will simplify the development and support maintenance activities when concerns are well separated: individual section can be developed and updated independently. The layered design is the embodiment of separate of the concerns. The known product of the layer design is the open systems interconnection (OSI) model: group communication functions into 7 layers, a layer serves the layer above it and is served by the layer below it.

In the Android technology, the development certainly embraces such design principle. For example: a simple Android application is shown below in Figure 2 and 3.

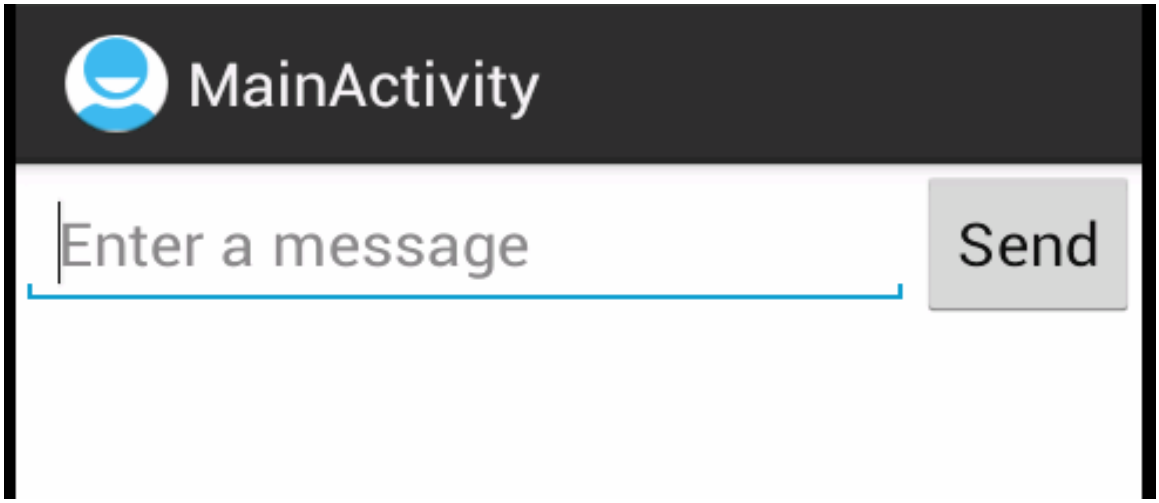

Figure 2 The Main Activity Screen

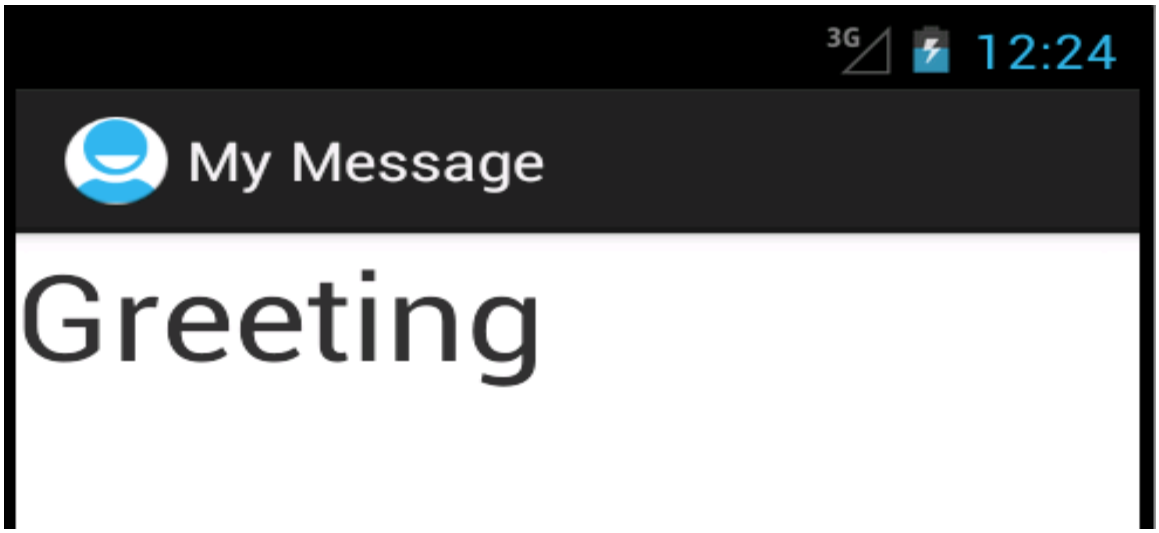

Figure 3 The Message Activity Screen

This application has two screens. The first screen, shown in Figure 2, is the start point of the program. It has a text field and a button. After the user enters a text and clicks the button, the program will change to the second screen, shown in Figure 3. This screen displays the message that the user entered.

The two screens' behavior are defined by two java files, MainActivity.java and DisplayMessage.java, located in the src folder. The screens' user interfaces are described by the two xml files, activity_main.xml and activity_display_message.xml, located in the layout folder. This separate of concern will support the flexibility of the program. If the interface needs some modification, for example, the "Send" button's location changes, but the program behavior stays same. Then we only need to change the first screen's xml file activity_main.xml.

The activity_main.xml is shown below. This file uses the linear layout, the text field and the button are placed horizontally in the screen.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    android:orientation="horizontal" >

        <EditText android:id="@+id/edit_message"
            android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />

        <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage" />

</LinearLayout>
```

4 Advanced Object-Oriented Concepts

The Android development heavily involves the use of libraries and event-driven programming. Some advanced object-oriented concepts including inheritance, polymorphism, interfaces, exception handling, streams and collection, are well practiced. Students are learning GUI programming and Android API during the development process. Example: the source code below is the MainActivity.java from the simple

application described in previous sections. It extends Activity class, and overrides several methods.

```
package com.example.maoandroidapp;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends Activity {

      public final static String EXTRA_MESSAGE =
"com.example.MaoAndroidapp.MESSAGE";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

    /** Called when the user clicks the Send button */
    public void sendMessage(View view) {
        // Do something in response to button
      Intent intent = new Intent(this, DisplayMessageActivity.class);
      EditText editText = (EditText) findViewById(R.id.edit_message);
      String message = editText.getText().toString();
      intent.putExtra(EXTRA_MESSAGE, message);
      startActivity(intent);
    }
}
```

More challenging topics such as multithreading, networking, XML and graphics are able to be incorporated in complex projects by taking advantage of specific benefits of mobile devices, such as built-in cameras, GPS, networking and sensors measuring touch, sound, acceleration and orientation.

5 Conclusion

Mobile computing has gaining its momentum in computing education due to the increasingly widespread use of mobile devices. We hope to make clear use of this technology to showcase computing concepts in new ways that are more effective and more engaging. On the course design, rather than just focusing on programming using new technologies, we review software design principles related to the Android app structure. Through developing apps in the real-world context, the learning potential will be fully realized and will advance students' proficiency in programming with advanced object-oriented topics.

References

[1] Computer Science 2008, An Interim Revision of CS 2001(http://www.acm.org/education/curricula/ComputerScience2008.pdf)

[2] Android Developer's Guide. http://developer.android.com/guide/index.html

[3] Abelson, W.F., Collins, C., Sen, R. *Unlocking Android – A Developer's Guide*. Manning Pub. April 2009.

[4] Victor Matos, Rebecca Grasser, Building Applications for the Android OS Mobile Platform: A Primer and Course Materials, *Journal of Computing Sciences in Colleges*, Volume 26 Issue 1, pp: 23-29, October 2010

[5] Derek Riley, Using Mobile Phone Programming to teach Java and Advanced Programming to Computer Scientist, ACM Special Interest Group on Computer Science Education SIGSCE 2012, pp:541-546. Feb. 29-March 3, 2012.