Development Environment for Android Application Development: an Experience Report

Robi Grgurina, Goran Brestovac and Tihana Galinac Grbac
University of Rijeka,
Faculty of Engineering
Vukovarska 58, HR-51000 Rijeka
e-mail: robig@riteh.hr, gbresto@riteh.hr, tgalinac@riteh.hr

Abstract – In the last decades we have witnessed an enormous increase in the end user acceptance of mobile communications. The appearance of mobile platforms based on the open source software has rapidly increased the interest into mobile applications development. In this paper, we present an approach to the Android mobile phone application development that is based on an open source software and open source development environment. The work presented here is the outcome of a student project. We outline the experiences and lessons learned.

I. Introduction

The mobile application development is an excellent choice for a beginner software engineering project that aims to introduce students to elementary development process activities such as design, implementation and testing. The reason is its low complexity, great opportunity for innovation, and huge market interest into new and innovative mobile applications. Due to its simplicity, the students can easily, and in relatively short period, get acquainted with elementary development concepts, techniques and resources that can be combined to produce mobile applications.

Among the variety of mobile platforms, the number of developed applications that are based on the Android operating system (OS) is increasing. Android is Google's open source mobile software environment that consists of Linux based operating system [1], [2]. The application development uses Java programming language and the virtual machine that optimizes the usage of memory and resources. This is particularly important for mobile applications. One of the reasons of increased interest into Android development lies in the existence of free of charge and open source development environment, such as Eclipse, with rich toolset and a number of interacting possibilities [3]. Moreover, only basic programming skills are required to start developing simple applications for Android mobile phone operating system. Beforehand we had experience in C and a little bit in C++ programming language.

Motivated by all these reasons, we have chosen to study the Android mobile phone application development and Eclipse Integrated Development Environment.

In this paper we report our experiences gained in the student project developing an Android mobile phone application. Our application has been developed using the Eclipse Integrated Development Environment (IDE) with Android Development Tools (ADT) plugin and Android Software Development Kit (SDK). Some useful additional plugins [3] like Check Style, Find Bug State Analyzer with Aware and Eclipse Metrics plugin have been integrated into Eclipse IDE as additional support to reduce the number of coding errors during the development. The application testing was performed by the Android emulator [3] and the Robotium test framework [4] during the development while the finished application was tested on the real device using "The Perfecto Mobile Handset Cloud" service [5].

The rest of the paper is organized as follows. The Android operating system is explained in Section 2. Our project, along with the development and testing strategy, is described in Section 3. The development environment that was used for the Android application development is described in Section 4. Experiences gained applying the proposed development and testing strategy is presented in Section 5 as the conclusion of this paper.

II. Android operating system

The overall Android system architecture, as described in [2], is shown in Figure 1. There are four main layers, each providing services for the layer above and using services of the layer below.

Android OS is built upon Linux kernel, which allows Android to be ported to a wide variety of platforms by providing hardware abstraction layer for Android. It is used for memory management, process management, networking and other operating system services.

The Android native libraries lie above the Linux kernel. They are written in C or C++, and compiled for the particular hardware architecture used by a mobile phone. These libraries provide window management, 2D and 3D graphics, media codecs, SQLite database and a browser engine.

Android runtime also sits on top of the kernel. It consists of the Dalvik virtual machine (VM) and core Java libraries. Dalvik is a virtual machine specially designed for low memory requirements that allows multiple VM instances to run at the same time. All code is written in Java.

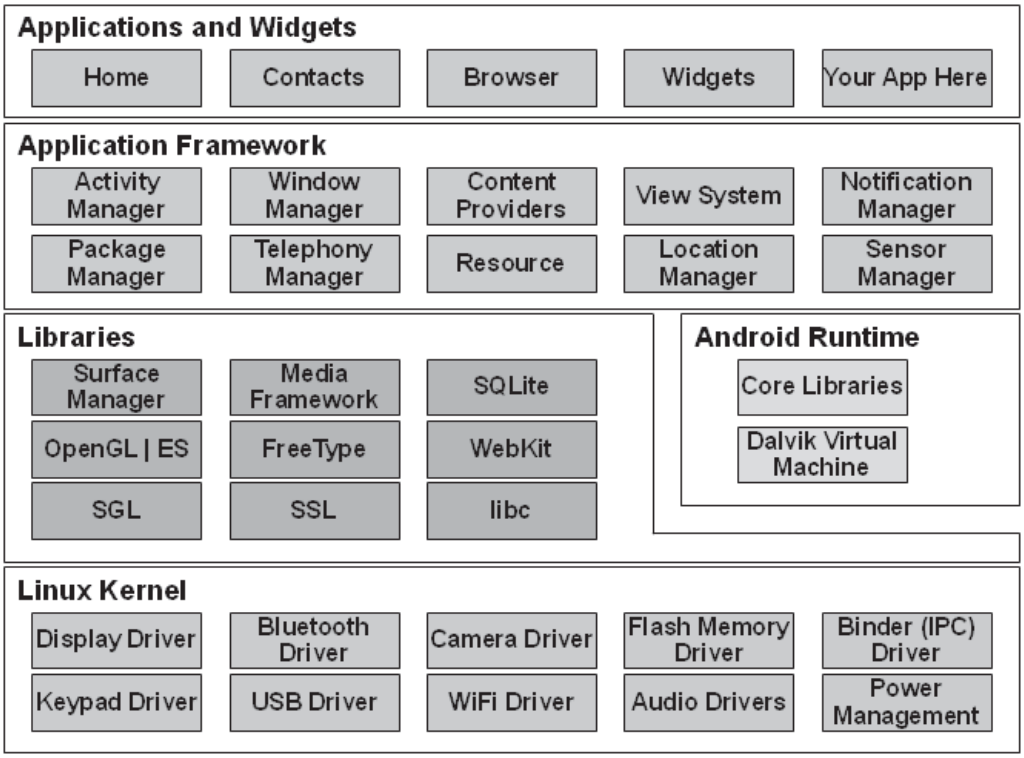

Figure 1. Android architecture

Application framework layer lies above the native libraries and the Android runtime. It enables the control of the lifecycle of the application, shares data between applications, provides location using the Global Positioning System (GPS) and manages notifications.

The uppermost layer contains applications and widgets. Applications are programs that interact with the user and usually take the whole screen, while widgets control only small rectangle areas within the user's home screen.

III. Project description

We worked on this project during the first semester of the academic year 2010/2011. The aim was to develop an Android application of our own choice.

We decided to develop an application that will help to expand user's English vocabulary. Our plan was for one person to develop the Android application, and for the other to develop corresponding test application. The development environment and tools are selected from the freely availabe tools.

A. Development process

The requirements were specified during the brainstorming session. The development of the Android application and the corresponding test application were run in parallel by two students. During the development there were weekly meetings at which we presented to each other the progress of the project and the encountered problems. Possible solutions were discussed, and we defined the tasks to work on until the next meeting. Meetings were held until all problems were solved.

B. Application

The application we were developing in the student project is an interactive learning game that is supposed to help players to expand their English vocabulary. This general idea is explored during the brainstorm session that resulted with a list of requirements. The list of requirements has guided our development and test process.

1. Application must be accessible through the main phone menu.

2. The main user interface provides options to play, change difficulty settings, add new words, and exit the application using buttons.

3. Pressing the "Provjera znanja" (eng. knowledge evaluation) button opens new interface which provides user the game case that consists of one Croatian word, and three offered English words, with only one of then the correct translation of the Croatian word.

4. Clicking one of the English words, the user is informed about correctness of the answer and the current

result state. At the same time, new game case with one Croatian word and three English words appear.

5. Pressing the phone back button the interface with the game case closes and the user is back at the main user interface.

6. Clicking the "Postavke" (eng. Settings) button opens new interface for game settings with three buttons that enable to among three difficulty levels: “lagano” (eng. easy), “srednje” (eng. medium) and “tesko” (eng. hard).

7. Pressing the phone back button the interface for game settings closes and the user is back at the main user interface.

8. Pressing the "Rječnik" (eng. dictionary) button opens new interface for dictionary extension with two text fields and three buttons. In the first text field the user enters a Croatian and in the second field an English word and choosing one of the three buttons, which present the difficulty level, the word is inserted into the database. Maximum length of words is set to 20 characters.

9. Pressing the phone back button the interface for dictionary extension closes and the user is back at the main user interface.

10. The "Izlaz" (eng. exit) button closes the application.

C. *Testing strategy*

A testing strategy has been proposed to define testing activities that we use within our project. The main steps of the testing strategy are listed below.

Firstly, already during the development process (including the development of both, the application and the test application), the compiler was continuously used to identify and eliminate inserted coding errors, like for example misspelled words. The compiled code was tested on the virtual machine.

Then, in early fault detection phase the aim was to identify errors that are against the coding standards. For that purpose a number of tools for automated checklist based testing can be used, such as for example [6].

At the end of the development process, the developed code is verified to check whether it behaves and performs as intended. The test application has to execute on the Android application aiming to verify the functionality described by requirements. All misunderstandings are discussed between the tester and developer.

Finally, after successfully passing all the previous testing steps, the final testing step according to the testing strategy is to install and test the Android application on a real device using the Perfecto Mobile web service [5]. The main purpose of this testing stage is to verify the application in the real environment, and on several different mobile phones.

IV. Development environment

Eclipse is an open source software development environment that we used in our project [7]. It comprises of an integrated development environment (IDE) and an extensible plugin system [3]. In the project we used Eclipse Helios release and several plugins that we present in the sequel. We developed both on Windows and Linux operating system.

A. *Android SDK*

Eclipse Integrated Development Environment (IDE) is the only Integrated Development Environment which is used for Android development and in that way it makes development more quicker and straight forward. Of course, Integrated Development Environment is not necessary for developers, but it is recommended by most developers beacuse of its simplicity.

Every Android application has its own Linux process and each process has its own virtual machine. Thus, every application has its own unique ID and it is running isolated from all other applications. The applications are formed from activities. Activity can be considered as a specific window inside an application. It is designed to do one specific task by interacting with the user. One application can contain different activities for each part of the program. For example, our application has one activity for playing, another activity for game settings, third one for entering new words etc. With services, Android has an option to handle multitasking.

A great example of a Service is a music playback that can be running in the background at the same time when the user is playing with our application. Our application has one database created with a SQLite. User access to the database with a query. All row that matches to his query description are returned as a type of a pointer (Cursor). The application have an Adapter class that handles specific calls to the database. Content Providers makes possible to access information between applications. As mentioned, each application has its own private sandbox and private database. Content Provider needs to be implemented by the application developer. Manifest is a kind of a security implementation which makes sure that application doesn't have parts that isn't publicly known.

B. *ADT Plugin*

One of the most popular plugin for development of Android mobile applications in Eclipse is the Android Development Tools (ADT) plugin developed by Google. The main purpose of this plugin is to allow greater ease of use to Android developers working in Eclipse environment. With this plugin developers can quickly and easily create new Android projects, debug their applications, add components based on the Android Framework Application Platform Interface (API), etc. It is not advisable to use text editor if you are developing large applications which contain a large amount of code because text editor doesn't have an option to highlight misspelling or lack of semi colon.

C. Android Emulator

Android emulator is included in every Android SDK. Android emulator is a virtual mobile device that runs on user's computer. With emulator it is possible to prototype, develop and test Android applications and there is no need to use physical device.

It mimics most of the features of a mobile device. The user is provided with a variety of navigation and control keys, which can be pressed using computer mouse. It is possible to generate application events with a keyboard. There is also a screen that displays user's application.

Android emulator supports Android Virtual Device (AVD) configurations. The AVD is an emulator which contains the specific Smartphone operating system. In AVD we specify the Android platform we want run on the emulator, as well as the hardware options and emuator skin files that we want to use. With AVD user can more easly test his application. When application is running on the emulator, it has an option to use the services of the Android platform like notify the user, store or retrieve data, play audio, access to network etc.

There are also few emulator limitations. In a release we used, there is no support for Bluetooth, USB connections, camera/video capture, nor support for placing or receiving actual phone calls and SMS/MMS communication.

D. Check Style Plugin

Check Style plugin [3] is another open source development tool to ensure that Java code adheres to a set of coding standards. With this plugin our's code was constantly checked for problems. We were notified of problems via Eclipse Problems View and source code annotiations. Using this plugin we could significantly improve code while coding, which can have the affect of discovering a potential defect in source code in a early stage of development cycle.

E. Find Bug State Analyzer With Aware Plugin

Find Bug State Analyzer plugin [6] is a powerful analysis tool which we used with the aim to avoid making dumb mistakes in our code. It comes with Aware plugin that displays alerts to the developer.

Table 1. Example report on Find Bug State Analyzer

Description	Resource	Rank./Sev.	Location
The class name android.test3.R$attr doesn't start with an upper case letter	R.java	0/1	Line11
The class name android.test3.R$color doesn't start with an upper case letter	R.java	0/1	Line 13
The class name android.test3.R$drawable doesn't start with an upper case letter	R.java	0/1	Line 18

Aware is an Eclipse plugin that gathers static analysis reports to generate a ranked listing of static analysis alerts and it is based on Find Bugs plugin. It collects alerts from Find Bugs plugin and displays them to the user, including a ranking and a severity.

Aware is also able to collect information about how a user uses Eclipse, such as what he is clicking on in Eclipse, which perspectives he is using etc. The user can filter alerts which he doesn't consider important. For example, if user sees alert that he wants to remove he can simply right click on the alert and choose option to Suppress Alert. After that, alert it send from Alerts view to Suppressed Alerts view. There is also an option to unfilter the alert by right clicking on the alert in the Suppressed Alerts view and selecting Unsuppress Alert.

We found this plugin as very helpful for programming. In most cases, mistakes had a normal priority type. Mostly our class names did not start with an upper case letter, ranking of mistakes was 0, and severity was 1. The example is presented in Table 1.

F. Robotium

After the main application was developed, our next task was to simulate a real users interactions.

Therefore, we were looking for a test framework that would help us to write powerful test cases that would emulate real users interaction. For that purpose we used a tool called Robotium [4].

Robotium is a test framework that makes easy writing powerful test cases for Android applications. With this plugin test case developers can write many types of test scenarios. It is possible to generate test case which spans accross multiple Android Activities. Robotium test case is able to do what a real user does: automatically move from one activity to the other one and click on anything clickable within application window. The main advantage of Robotium is the ability to develop powerful test cases, with a minimal knowledge of the testing application.

With our test application we created 19 test cases for our main application. We did various tests based on the user interactions like how the application would behave when the user clicks the button, enters new words or change difficulty level of playing. It was also necessary to test whether the result of playing behaves as we except. More precisely, when the user hits the correct/wrong answer or situation when the user clicks the Exit button would the application really shut down. Also, we needed to test interaction with the database, so with test application we were entering new words into dictionary database. If a test fell, status bar changed its color to red and we were notified about this event with an explanation. Then we would correct the error and repeat the test process until the status bar change its color to green which means that test passed.

G. Eclipse Metrics Plugin

We used this plugin to calculate various metrics for our code. With Metrics plugin [8] we could stay aware of the health of our code base. It is possible to export the metrics to Extensible Markup Language (XML) or comma separated values (CSV) format. There is also an option to export to HyperText Markup Language (HTML) form. Metrics can be considered as indicators of unhealthy code, i.e. codebase with many range violation warnings may

indicate that code needs to be refractored but no range violation warnings doesn't always mean that code is good.

To run metrics on codebase we needed to install Eclipse running under JDK 1.5 at least. It runs on Linux without problems. In order to display metrics about a project it was necessary to enable the project to calculate metrics. We needed to make sure we are in the Java Perspective and had our project open in the Package Explorer. Next task was to right click on a project and in properties window select Metrics on the left side, and check the Enable Metrics checkbox. For more information about this plugin please visit official web site [8].

Table 2 shows a lot of useful information that we obtained using the metrics plugin. We were able to see

Table 12. List of metrics from the Eclipse Metrics

Metric	**Total**
Number of Overridden Methods	0
Number of Attributes	20
Number of Children	0
Number of Classes	13
Method Lines of Code	322
Number of Methods	22
Number of Packages	2
Total Lines of Code	507
Number of Static Methods	0
Weighted methods per Class	66
Number of Static Attributes	44

how many classes, lines of code or overridden methods we have in our application. With 322 total method lines of code we come to the conclusion that our application is not complex. For example, if a method is over 50 lines of code we were suggested that the method should be broken up for readability and maintainability. Mostly Metrics plugin was warning us that the class name does not begin with a capital letter, but this warning had a low range violation.

We found very usefull a dependency graph view which is part of Metrics plugin. With this graph view we were able to view dependency connections among various packages and classes in our project. Our application is not very large, does not consist of a lot of packages, so dependency graph view is not complex.

In Figure 2 there is a dependency graph showing the dependency connections among various packages and classes in our project. The rectangle in the center is representing our project and rectangles connected to it are representing classes that our application uses and depends on.

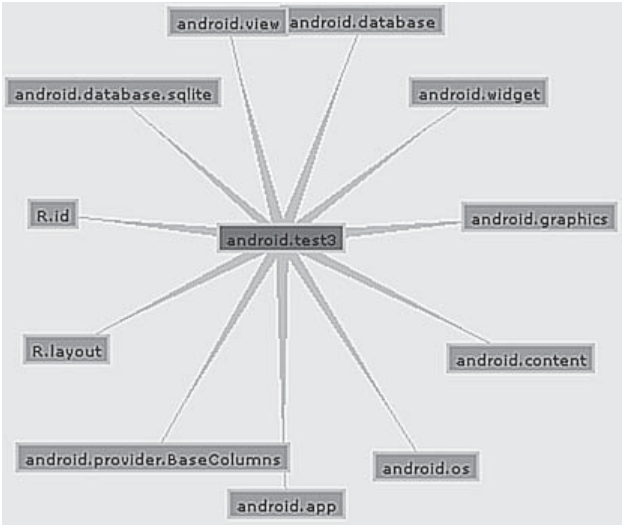

Figure 2. Dependency graph view

H. Test Mobile Applications on Real Devices

Finally, after the main application and appropriate test application were developed, our next task was to see how our application behaves on a real devices. Thus, we found a service called "The Perfecto Mobile Handset Cloud" [5], which enables testing mobile applications, websites or services on a multitude of real handsets. With this system the developer is able to access to real mobile devices via the web and control them as if he were holding them in his hands. Developer can quickly and easily operate the phone keys or touch-screen to control the device, install applications on the device and verify their application or service works properly and displays correctly on the phone's screen.

First of all we had to register for the service, select mobile devices that run the Android operating system and then install our application on it. Then we tested our application on a two android devices: a HTC Desire and on Motorola Droid mobile phone. Results were excellent, we haven't found any bugs and our application was working on both devices as was expacted.

V. Experiences and lessons learned

In this paper we have presented a student project developing an simple application for Android mobile phone. The focus of this work was on the processes and development environment that was based on open source software and we used within the project.

Before this project we have never developed mobile application and we had no experience at Java programming. In the beginning of the project we didn't know what kind of application we wanted to develop and we needed some time to decide. We were overwhelmed with possibilities.

System requirements were defined during the brainstorming session. During application development

and test we found out that some specifications needed extra clarification because everyone had a different interpretation of the specifications. Also we left out some details that could improve user experience like displaying current result.

This experience was very useful because we learned something new and got insight into the entire process of creating applications. We understand the importance of formal processes for effective teamwork as is for example in our case insufficiently defined requirements have slower our development and testing by increasing the otherwise unnecessary communication overhead.

The Eclipse IDE was choosen because there is official Google's ADT Plugin used for Android application development so we expected less bugs than on nbandroid plugin used by NetBeans IDE and we wanted to use several additional Eclipse plugins to reduce bugs. Its plugin system allows us to use several plugins which we found very useful. For example, our progress has been significantly accelerated by using Find Bugs State analyzer plugin [6] which immediately reported coding errors by highlighting the corresponding line of code.

With a CheckStyle plugin [3] we were able to improve the quality of our code early in the development cycle. Finally, with a Metrics plugin [8], we got an insight into the different metrics of our codes such as the total number of lines of code, number of packages and other metrics listed in Table 2.

For testing Android applications we used a testing tool called Robotium [4]. Robotium is very easy to use and helped us to keep very high quality of our application throughout the project lifecycle. With the support of Robotium, we were able to write system and acceptance test scenarios, spanning across multiple Android activities. goal was to see how our application will run on real mobile phones. For that purpose we found the service on the Internet called the Perfecto Mobile [5] that makes this possible. With Perfecto Mobile service we realized that our application works on real devices as we expected.

VI. CONCLUSION

The paper presents experiences and lessons learned from a student project which main goal was to gain basic software engineering skills, experience teamwork, understand the software development process and experience the Android development environment.

In the project a simple Android application is developed by a pair of students: developer and tester. All supporting tools used for development are open source tools and free of charge. Android platform and related freely available open source development environment have shown as excellent for the student project. In relatively short period of time, we were able to perform a simple student project and experience challenges of development process and working in team. Moreover, we experienced development environment and in sequel we present the lessons learned.

We chose Eclipse IDE because it uses official Google's ADT Plugin for Android application development in which we expected less bugs than on nbandroid plugin used by NetBeans IDE. Some bugs were encountered but they did not present significant problems, Till example after the creation of new Android project Eclipse found a lot of false errors, but after restarting the Eclipse IDE, false errors disappeared. We used several Eclipse plugins for bug detection. It is preferable to use more than one plugin to check the code because if one plugin misses some potential error, another might find it.

Although JUnit is recommended for testing Android applications we used Robotium instead because we found it easier to write test cases while the code is shorter and at the same time powerful enough for testing our simple application.

To develop Android applications it is not necessary to own a physical Android mobile device. We found "The Perfecto Mobile Handset Cloud" useful to test our application on a multitude of real devices.

REFERENCES

[1] Android Developers: http://developer.android.com/, Access Date: 6.1.2011

[2] Ed Burnette, Hello Android, Introducing Google's Mobile Development Platform, 3rd edition, Pragmatic Programmers, 2010, pp 30-35.

[3] Automation for the people: Improving code with Eclipse plugins, http://www.ibm.com/developerworks/java/library/j-ap01117/index.html, Access Date: 20.01.2011.

[4] Robotium, http://code.google.com/p/robotium/, Access Date: 23.12.2010.

[5] Test Mobile Applications on Real Devices, http://perfectomobile.com/, Access Date: 20.01.2011.

[6] FindBugs Static Analyzer and AWARE in Eclipse, http://agile.csc.ncsu.edu/SEMaterials/tutorials/findbugs/, Access Date: 22.12.2010.

[7] Eclipse Project, http://www.eclipse.org/eclipse/, Access Date: 06.02.2011.

[8] Software Metrics in Eclipse, http://agile.csc.ncsu.edu/SEMaterials/tutorials/metrics/, Access Date: 25.12.2010.