Software Requirements Design

for

Cyber Engineering Pilot

Version 0.1

Prepared by

Group Name: CSC 403 – Louisiana Tech University

Brett Clavier	Drew Gardner	Ali Khan
William Metcalf	James Poore	Rajan Sharma
Dawa Sherpa	Tyler Simmons	Drew Smithies
Adam Tannehill	Stephan White	Gerald Moreland

Instructor: Jean Gourd

Course: CSC 403

Date: February 3, 2012

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Alpha V 0.2	CSC 403 Pilot Group	Concept generation.	01/19/90

1 Introduction

1.1 Document Purpose

A cyber engineering pilot program is being developed to assess how the production program should be taught and executed. We will be developing 12 labs spanning two courses that will reinforce concepts taught through lecture and other means of teaching. We will discuss the possible lab topics and how to properly conduct the labs. We will also cover the requirements associated with this pilot.

1.2 Product Scope

The students will learn programming skills through weekly (or biweekly) labs. These biweekly labs are meant to give the students an introduction to basic and intermediate programming concepts. Their development environment will include the Android SDK using the Eclipse IDE and ADT plug-in. A Samsung Nexus S cellphone and Android device emulator will be used for debugging, program execution, and testing.

1.3 Intended Audience and Document Overview

This document is intended for the project manager to assess if our project plan is acceptable. In the section two, we will cover the perspective, possible topics, users, development environment, and assumptions. Section three will cover software and hardware interfaces and specific lab ideas. Section four will cover performance requirements and possible safety concerns.

1.4 References and Acknowledgements

"Android Developers Package Index." Internet: developer.android.com/reference/packages.html, Jan. 18, 2012 [Jan. 19, 2012].

Dr. D. Janzen. "Android App Course." Internet: http://sites.google.com/site/androidappcourse/, [Jan. 19, 2012].

2 Overall Description

2.1 Product Perspective

This is a brand new program that will be replacing the existing curriculum for the CSC 120 and CSC 122 classes. The program will also be a part of the upcoming Cyber Engineering major being offered by Louisiana Tech University starting the spring quarter of 2012.

~~2.2~~ Product Functionality

- Engage students in learning
- Reinforce concepts
- Topics will include, but are not limited to:
  - Objects and Classes
  - Class Definitions
  - Object Interaction
  - Grouping Objects
  - Designing Classes
  - Inheritance
  - Extensible, Flexible Class Structures
  - Graphical User Interfaces
  - Error Handling
  - Application Design

2.3 Users and Characteristics

The users will be undergraduate students of varying skill levels. Many of these students will only possess the programming skills taught by CSC 100. Other students may have already attended the old CSC 120 and 122 classes, but are taking them again due to the differing structure. There will also be at least two senior level students taking the course to assess the structure of the labs and lessons.

2.4 Operating Environment

For the students to succeed, they will need a computer, laptop or desktop, capable of running any modern operating system. The preferred OS is any user friendly Linux distribution, but Windows XP, Windows 7, or Mac OS X 10.5+ are also usable operating systems. The student's selected OS must be capable of running the most current version of the Android SDK and device emulator. Also, Google suggests using Eclipse with the ADT plug-in as the IDE. Google supplies a thorough tutorial on how to setup a proper development environment, including path variables, using Eclipse.

2.5 Design and Implementation Constraints

The main constraints that could be encountered during development include limited knowledge of the Android SDK, limited knowledge of the Android API, programming functions limited to the Android SDK, development time is only about a month, and future Android firmware updates that change or remove functions required in the labs.

2.6 User Documentation

For users to be able to complete the labs, they will need to follow the tutorials provided by Google on how to install the Android SDK, incorporate the tools into the system path, and how to configure Eclipse to use the Android SDK. They will also need unfettered access to the Android API also provided by Google. Other useful tools may include other SDK tutorials or sample application code.

2.7 Assumptions and Dependencies

Some key assumptions we are making include that the users will have limited or no programming experience, every user will have access to an Android device, and that every user will have access to a computer for programming the applications.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The minimum interface the students should use is the Eclipse IDE with the ADT plug-in. They will also need the Android SDK installed with the Android 2.3.x libraries installed (this may change to Android 4.0.x if the Android devices are updated to 4.0.x). Also included with the Android SDK is a device emulator that allows the student to run an application in a virtual device before attempting execution on a physical device.

3.1.2 Hardware Interfaces

Each student should have access to a university provided “Samsung Nexus S” cellphone running (as of writing this) Android 2.3.x.

3.1.3 Software Interfaces

Interface with the devices, Android 2.3.x (possibly 4.0.x) will be used in conjunction with the Eclipse IDE. Linux, Windows (XP or higher) or Mac OS X 10.5+ are required to use this software.

3.2 External Design Specifications

These two classes will include a total of 12 lab assignments (6 introduction and 6 intermediate). The introduction labs will be divided as follows:

(1) An example program will be supplied to the students. After explaining the functions to them, they will be required to edit it so that it preforms some specified action.
(2) External method calls will be covered, including passing objects as parameters and object diagrams
(3) Loops and arrays
(4) Using API calls and proper documentation
(5) A buggy program will be provided to the students. They will be required to debug it to get it to work.
(6) The final lab will require them to write a program from scratch which incorporates all of the previous labs.

The intermediate labs will be divided as follows:

(1) Inheritance, polymorphism, and subtyping
(2) More inheritance and method overriding/overloading
(3) Supply a basic Graphical User Interfaces (GUI) and have the students modify it
(4) Build simple GUI application from scratch
(5) Supply a broken program and have them debug and add proper error checking/handling
(6) Build a program from scratch incorporating the previous labs.

Initially, we are going to focus on three labs from both the introduction set and intermediate set of labs. The synopses of these labs are as follows:

1. Loops and Arrays – Play a Tune

   This lab will teach students about loops and arrays. They will create loops that will play a sound. The playing of the sound will reinforce how the loops are implemented and give them audible feedback to learn from. Arrays will be used to hold information and will be incorporated into the lab as well. This will be a very basic lab and will not require much knowledge of the Android framework to complete.

2. Using API Calls - Alarm System

   The objective of this lab is to teach students how to use an API. This will be accomplished by implementing an alarm system. Students will be required to search the Android API for classes that listen for and play sound, and then import these classes. A simple example of a program using an API call will be given to the students so they can better understand how this is done. Once they find the methods in the API that allow them to listen and play sound, the alarm system will be implemented. The phone's mic will listen for sound, and if it receives sound over a certain decibel, it will play an alarm.

3. Comprehensive Project - Chromatic Tuner

   The objective of this lab is to provide a challenging problem that incorporates all of the elements learned in previous labs. Students will be required to implement a basic chromatic tuner given the bare minimum of starting resources. At the very least, these resources will include the requisite fast Fourier transform function in Java and a description of its usage. The end result will be a program that listens to a tone through the Android's microphone, identifies the nearest representative frequency on a predefined scale, and indicates to the user whether the tone is high, low, or close enough compared to the frequency.

4. Polymorphism – Animals

   The objective of this lab is to teach students about Inheritance, polymorphism, and subtyping. This will be accomplished by building an Animal sound system. The lab will teach the idea of inheritance and polymorphism by playing different sounds for different selection of animal pictures. To implement this lab, the pictures will be provided and the students will have to map each picture with the sound utilizing the idea of inheritance and polymorphism. A preview video of the final application will be provided.

5. GUI – Calculator

   The objective of this lab is to teach students about Graphical User Interfaces (GUI's). A pre-made GUI will be provided to the student in the form of a calculator application, but the GUI will lack all functionality. The student's objective is to implement code to provide proper function to the existing calculator GUI. By doing this the student will be required to take note of how the GUI functions in order to give it functionality. Being that a calculator is centered around a GUI for function, it will provide a solid foundation to the student on

6. Comprehensive Lab – GPS Scavenger Hunt

   The lab will incorporate all the topics covered in previous labs. The students will create a GPS guided scavenger hunt app. The Near Field Communications sensor will also be used to gather information. The students will build this app from scratch to demonstrate their knowledge of the curriculum topics.

A sample lab will be attached to this document that will show how we plan to format the lab documents. Below, each section of the lab document is described.

1. **Introduction** - This section will give a brief description of topics and concepts to be learned. A general overview of what the lab will contain will also be given.
2. **Objective** - This section will go into more detail about what concepts should be learned and goals to be accomplished.
3. **Activity** - In this section, we will explain the program that will be written. Students will be given a guide on what they will be implementing.
4. **Conclusion** - This section will give a brief summary of what the student should have learned, and require them to answer questions that reinforce the concepts taught.
5. **Deliverables** - This will be the programs the students write.

4 Other Non-functional Requirements

4.1 Performance Requirements

The labs will be approximately 2 hours long and will allow the students between one to two weeks (or more) depending on the relative complexity of the assignment. If a student does not complete a lab assignment, he/she will fall behind and possibly be unable to complete the following assignment.

4.2 Safety and Security Requirements

Some possible safety concerns are mainly attributed to the Android device and include:

- Physical damage to the device
- Theft of the device
- “Bricking” the device’s software due to rooting or flashing custom firmware

5 Internal Design Specifications

5.1 Project Management

We will be using the agile process model under the consideration that the requirements of the project are not concrete enough to definitively lay down a path from start to finish. In addition, this model suits our team organization, which is primarily peer-to-peer and composed of small groups. Since the project can be decomposed into several smaller projects (the labs), the agile process model affords us the flexibility to meet the ever-changing expectations and requirements that apply to each lab.

5.2 Project Leader

Our project leader is Drew Gardner. He will oversee meetings and delegate responsibilities.

5.3 Group Organization

We divided into six groups, each group in charge of one lab. The groups are as follows:

1. Loops and Arrays - Play a Tune
    1.1. White and Moreland

2. Using API Calls - Alarm System
    2.1. Simmons and Dawa

3. Program from Scratch - Chromatic Tuner
    3.1. Gardner and Tannehill

4. Inheritance, Polymorphism, and Subtyping - Shapes / Animals
    4.1. Sharma and Khan

5. GUIs - Calculator
    5.1. Poore and Clavier

6. Program from Scratch – GPS Scavenger Hunt
    6.1. Smithies and Metcalf

5.4 Configuration Management

Configuration management will be accomplished with the aid of GitHub, a free online code repository and version controller. With GitHub, everyone can have access to the same files in the same location, and there does not have to be any worry about working on a file that is out of date or isolated from coworkers' contribution. In addition, the version control software built in to GitHub ensures that valuable work is not lost when changes are made, such as accidental file deletion or an idea that did not work out. GitHub also provides the means for team members to branch from the same initial source, work separately, and then merge back together without unnecessary disruption.

5.5 Risk Management

We acknowledge that our work does not exist in a vacuum; it has a deadline and people that need to utilize its results. Therefore, steps should be taken to ensure that completion of part or the entire product does not fail. One minor way in which we addressed this was to assign a minimum of two people to each lab. Ultimately, the problems present in the labs are solvable by one person, so having at least two people ensures that if one becomes incapacitated, the other can pick up the slack. If both become incapacitated, then someone can be pulled off of a lab with two people so that the equilibrium of at least one person per lab is maintained. Since the labs are being implemented in two phases of six labs each and there are twelve team members, this indicates

that the minimum work force required for the project to move forward is six people. With a buffer of half the team, only a significant problem could force the project to a halt.

There is also the risk that the customers, the students that are to follow and the professors that are to teach, will not like what we offer. Such a situation would be costly, wasteful, and difficult to recompense our losses, not to mention the damage it might do to the program in turning away underclassmen. We hope to avoid such a scenario altogether with regular if not frequent feedback from the professors. The students can not, of course, offer their opinion until it is too late, but it is possible to offer a trial to students similar to our target audience. The pilot course offered in the spring does just that, acting effectively as a closed beta. The feedback from the pilot will provide valuable direction for the concurrent development of the final labs.

5.6 Quality Assurance

Each group's work will be evaluated by other groups at the end of the implementation process to ensure it is satisfactory and meets all requirements. Also, during each team meeting, each group will be required to give an update on the status of their progress. In addition to our in-group evaluation, the labs will also be evaluated by the web team, the other half of our class designing the Computer Science website. This will happen during the two week buffer period. The last evaluation will come from the actual Pilot class itself. Students will be required to write a one page reaction paper after each lab describing the effectiveness and quality of teaching of the required topic. Each evaluation serves to ensure that what we offer is not of poor quality. Negative feedback will be taken into consideration with a revised version of the relevant lab.

Black box testing is desirable as a means to ensure that our labs do as we claim, thus reducing the chance of faulty directions. White box tests are not as applicable since at this stage merely the basic concepts of programming are being taught with little consideration for the efficacy of the method. Since the minimum standard our own work must meet is that of the student (to show as an example of the correct solution), there is less emphasis on providing the best solution, especially since the best solution might be beyond the students' understanding.