Analysis, Design & Testing

SEG2012GP9

|  |  |
| --- | --- |
| Seabrook, Edward J F | [ejfs1g10@ecs.soton.ac.uk](mailto:ejfs1g10@ecs.soton.ac.uk) |
| Elliott, Kristian | [ke1g10@ ecs.soton.ac.uk](mailto:ke1g10@soton.ac.uk) |
| Yu, Brian | [by2g10@ ecs.soton.ac.uk](mailto:by2g10@soton.ac.uk) |
| Mariani Elola, Oscar N | [onme1g10@ ecs.soton.ac.uk](mailto:onme1g10@soton.ac.uk) |
| Chan, Yun Chak | [ycc1g11@ ecs.soton.ac.uk](mailto:ycc1g11@soton.ac.uk) |

# Notes to the Authors:

* Max 12 sides A4 in total for this report, excluding "wrapper" pages (e.g., title page, contents page, document control page) and appendices

# Document Control

|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Changes** |
| 1 | ejfs1g10 | Initial document |
| 2 | onme1g10 | Added conceptual model of HCI |
| 3 | ke1g10 | Added the approach taking |

# Table of Contents

Notes to the Authors: 1

Document Control 2

Table of Contents 3

1 Approach Taken 4

1.1 Technical 4

1.2 Modelling 5

2 Analysis Documentation 6

2.1 Requirements Analysis 6

2.2 Design Analysis 6

3 Testing 7

3.1 Test Plan 7

3.2 Partial Test Report 8

4 HCI 9

5 Screenshots 10

# 1 Approach Taken

## 1.1 Technical

1. The project specification requires that we work in the Java programming language; this works out well as we have all had experience working with Java and are familiar with it.

1. Eclipse has been chosen by our group as out preferred development environment. The reason for this choice is that it is a good, extensible Java IDE (Integrated Development Environment) with features such as autocomplete that help us write code faster than in a standard text editor. Eclipse is also cross platform which is a great benefit since we use a range of Operating Systems within our group. We also all have experience using Eclipse.
2. Google’s Window Builder Pro Eclipse plugin shall be used to speed up the creation of Graphical User Interfaces for the program.
3. We shall be using Microsoft Word 2010 to produce the formal documentation and reports for the project. Word is good at handling medium sized documents and is easy to style to the house guidelines. We shall however submit our reports as PDF files to ensure good compatibility.
4. Google Docs will be used as scratch space for rough notes and ideas, and also to circulate agendas before meetings. Google Docs was chosen because it has been designed to facilitate collaboration. Google Docs also offers a spread sheet application that we shall be using to log out time spent.
5. We have chosen to use the ECS ugForge server to host our version control repository. We have attempted to make regular commits of our work to ensure high levels of transparency on the work being done, and allow rollbacks if a major error is found in our work.
6. Our Gantt charts shall be produced using [www.smartsheet.com](http://www.smartsheet.com), The web based service produces clean, maintainable charts. Using this software will allow us to monitor our progress and decide which tasks to perform next. Frequent updates to the chart will ensure it reflects the real world.
7. Many types of diagram will be used to help us understand various aspects of the project:
   1. Use Case diagrams, produced in visual paradigm, shall help us identify and comprehend all of the use cases from the requirements.
   2. Class diagrams, produced using PlantUML (a text based UML tool), allow us to break the software into classes that can be coded.
   3. Sequence diagrams to see how these classes are meant to interact within the system.
   4. Wireframes of the user interface will be produced using Balsamiq to help us develop a user friendly user interface.

## 1.2 Modelling

To produce good quality, extensible software, a number of different abstractions and architectural styles have been adopted throughout our application:

1. The project specification mandates the use of Model-View-Controller architecture. This architecture separates the classes of the system into three different categories.
   1. The Model classes are concerned with the problem domain – all the business logic and calculations for the application are stored in the Model classes.
   2. The View classes perform all of the drawing to the screen. In java these are the Swing classes that produce the user interface. There will be classes for the various windows used in the application. Most of this code shall be generated using Google Window Builder Pro. There shall also be classes that extend the functionality of the standards Swing components to achieve encapsulation of graphical elements; one example of this is the RunwayView class that shows a visualisation of the runway.
   3. The Controller classes interact with both the View classes and the Model classes. In essence, the Controller tells the Model to update and lets the View know that the Model has been updated so it can redraw the data.

We have chosen to keep the three groups of classes very separate, splitting into pairs to develop them, this allows the program to be more extendable, as the coupling is very low.

What other patterns have we used?

Iterator.

# 2 Analysis Documentation

Analysis documentation: This section should include as appropriate class diagrams, statecharts and sequence diagrams, and explanatory narrative. This narrative is important - you need to make it easy for your examiner to see what's going on. Assume he knows nothing about the design of your system. The narrative should include a list of your significant class & method names and definitions.

**Proposal:**

## 2.1 Requirements Analysis

UML such as stakeholder diagram

## 2.2 Design Analysis

UML such as class diagram

# 3 Testing

## 3.1 Test Plan

Detailed test plan including numbered tests, test scripts, expected results. To include the demo UAT.

## 3.2 Partial Test Report

Partial test report including a good representative selection of test results, including UAT. We understand that it is difficult to complete your testing by this date but expect coverage of at least 75% of plan. Ideally, this and the test plan will be combined in tabular format.

# 4 HCI

***4.1 Conceptual Model.***

This software product will provide visual and graphical tools to recalculate and present revised runway parameters, visualizations and summarized calculations to the customer, given and obstacle located in it.

This team will take advantage of the widely known components of Java’s Graphical User Interface libraries and their high affordance and ease of use (e.g. drop down menus, menu bars, buttons, text areas, etc.) and allowing focus on those considered as the main usability goals.

While designing this system, the following usability goals will be regarded as most important:

* ‘Effective to use’; the aim is to build a set of tools that effectively ‘solve the problem’.
* ‘Safe to use’; the aim is a system with low error rate and easy recovery.
* ‘Good utility’; the system will count with a very appropriate set of functions to aid the customer when dealing with obstacles in a runway.

The system assumes a set of users familiarized with the concepts that they will be exposed to, including CAA’s (UK Civil Aviation Authority) rules and regulations, runway redeclaration, and runway parameters, including TORA (Take-off run available), ASDA (Accelerate Stop Distance Available), TODA (Take-Off Distance Available) and LDA (Landing Distance Available).

With the aid of this software product, rapid indication of the effects of an obstruction can be obtained and used to decide if operations in the runway can continue and if performing the manual calculations is worthwhile.

# 5 Screenshots

A small number of screenshots to enable your examiner to make the link between your demonstrated and your paper submissions (as an appendix).