Onerous Android Application

Architecture/Design Documentation

Table of Contents:

1. Introduction
2. Specification
3. Logical View
   1. High-Level Design
   2. Mid-Level Design
   3. Low-Level Detailed Design

1 Introduction

This document describes the architecture of the Onerous App developed by the Modelling and Simulations Group of Cranfield University based at the Defence Academy Shrivenham. The Onerous App will be used to teach and demonstrate students about stochastic discrete event simulation and their use in operational analysis. It models helicopters stationed at the frontlines with engines that need to be regularly maintained.

The purpose of this document is to describe the architecture of the Onerous App clearly and accessibly to meet a few requirements:

1. Reduce time taken for future developers/programmers to understand the code at a level needed to be able to maintain, modify, enhance and perhaps even port the app to another platform such as iOS.
2. Allow students to explore and understand how the simulation is built and how it works.
3. Provide multiple views to support various different specialised interests in accordance with IEEE std. 1471.

What this document will not do is explain the Android or Java framework. Understanding of either should not be necessary in understanding the architecture of the application. However, competence with both Java and Android will be necessary for code maintenance or modification.

It will not detail how the individual methods or functions operate but will summarise their jobs and input/outputs. If further technical detail concerning the implementation of individual methods and functions are required, the commented source code can be found in the project files.

2 Specification

The Onerous Android App based on the original Onerous V2.0 desktop application, which will be referred to as Onerous 2008. Onerous 2008 is a desktop application built using Microsoft Visual Studio 2008 using the Visual Basic language.

The objective was to port Onerous 2008 so that it could run on tablets running the Android OS. The specific model that was available at the SSEL was the Samsung Tab 2 10.1 running android version 4.1.1 (16 Jellybean). All functionalities had to be retained and the app UI had to be updated to one that is geared towards tablet usage with a touchscreen as opposed to Desktop usage with mouse and keyboard.

For further information on the specification and details leading to the creation of the original Onerous 2008, file OA 1229 and the presentation file FMS\_SSE\_16\_7\_Onerous\_Responsibility can be referred to.

System Behaviour overview

The system behaviour will be explored in more detail in the use case view section. However, this section aims to give a brief summary of how the app should behave and what it functions it should be fulfilling to lay the foundations for better understanding of the app.

This a repeatable, three-step process is a typical way a user could navigate the app when operating it.

Variables & Data

*Read*

*Read*

*Write*

*Read*

*Write*

Variables Tab

Simulation Tab

Results Tab

Below is a brief summary of what role each of the three tabs fulfil in the overall operation of the app.

1. Variables Tab: allows the user to view the current simulation variables and to change them if desired.
2. Simulation Tab: allows the user to run the simulation with the set variables. They can choose to view the graphics and animations or choose to toggle them off for an instant run. The simulation uses the user’s settings to run and as it runs it generates the statistics and the graph data and stores them.
3. Results Tab: allows viewing of the statistics and graphs with data generated from the most recent simulation.

**3 Logical View**

The logical view describes the components of the app and how they work together to provide the overall functionality of the application. The logical view will begin at a high level view and work its way down to low level. High level will explain and show how the major components work together at an architectural level and will then progress towards more detail by showing smaller components and then eventually to individual methods and functions, but any further detail will require a look at the source code and the comments included with it.

**3.a High**

**Class references**

**ActivityDistribution**

The ActivityDistribution class is a user defined type that holds values that will be used to calculate how long each ‘activity’ (e.g. engine removal, engine refit, bad engine transit etc.) takes.

**EventData**

The EventData class is a user defined type that is used to hold two bits of data: the time of an ‘event’ and the ‘kind’ of event.

**Events**

The class Events contains the methods used for performing the actions required when a certain scheduled event occurs.

**MainActivity**

The MainActivity class is the very first class that launches when the app is launched. It is responsible for initialising the UI view and setting up the tabbed fragments that will hold the majority of the UI elements.

**PlaceholderFragment**

Tabbed placeholder fragment that can hold the place of a tab in a tabbed activity.

**ResultsData**

The ResultsData is a user defined type that is used to hold data and statistics when the simulation runs.

**ResultsFragment**

The ResultsFragment class initialises the UI elements of the ‘Results’ tab, and contains the methods required for populating the results table and the graph with the appropriate data.

**SimAnimations**

The SimAnimations class contains all the methods for performing the various animations when visually displaying the simulation running.

**SimMethods**

The SimMethods class contains much of the backbone for actually simulating the scenario.

**SimulationFragment**

The SimulationFragment class initialises the UI elements of the “Simulation” tab.

**Stats**

The Stats class contains the methods used when generating the simulation run stats.

**Variables**

The Variables class contains all the global variables that the application needs to access from multiple other classes. Many of these global variables are also the variables that control the simulation run settings.

**VariablesFragment**

The VariablesFragment class initialises the UI elements of the “Variables” tab which allows the user to change many of the simulation settings.