TEST PLAN

Simulated computing system

December 22, 2017

António Pedro Araújo Fraga

Student ID: 279654

Cranfield University

M.Sc. in Software Engineering for Technical Computing

Contents

Test Plan Identifier

Simulated Computing System, Fraga

References

This test plan is based on the **IEEE 829 format**, regarding to a test plan outline.

Introduction

This system will be developed under two modules, **Software Testing and Quality Assurance** and **Requirements Analysis and System Design** at **Cranfield University**.

It is supposed to simulate a computer system which runs jobs of several sizes. The jobs last between **one second** and **thirty eight hours**, assigning them with a category of **Short**, **Medium**, **Large** or **Huge**. The system shall be capable of scheduling a job running time on a basis of a **First Come, First Served** methodology, analysing what is the correct time to run a job every time a submission is made. The simulation must generate a set of informations regarding the pretended outputs. These outputs are dependent of inputs values that will be established in the beginning of the simulation.

The application is going to be used by the IT department, modelling the behaviour of a real computing platform and exploring alternative accounting strategies.

The system shall attend functional and non-functional requirements. Most of functional requirements shall be tested at the **Unit & Integration level**, whereas non-functional requirements shall be tested at a higher level.

Features and Functions to test

- Coding standards

- Compatibility
- Functional
- Reliability

Approach/Strategy

Coding standards

The developed code shall follow the coding standards written in the **Software Requirements Specification** document. In order to achieve this, it is going to be used a checker as an IDE plug in, called **cppchecker**.

Compatibility

The system must be capable of running in different environments, therefore, the system functional requirements shall be tested in different machines, running different **Operating Systems**.

Functional

Functional requirements must be tested with Unit Tests, making use of a proper tool (**Catch**). These requirements must also be tested on the **version control** system, making use of **Travis**. This implementation will allow **integration testing**.

Reliability

The system shall not crash when put under **Stress Tests**. Extreme situations must be tested, expecting correct functioning and results.

Item Pass/Fail Criteria

Every feature must be tested and approved.