McMaster University

CAS 4ZP6

Team 9

Capstone Project 2013/2014

Porter Simulation

Test Plan Revision 0

Authors:
Vitaliy Kondratiev - 0945220
Nathan Johrendt - 0950519
Tyler Lyn - 0948978
Mark Gammie - 0964156

Supervisor:
Dr. Douglas Down

February 4, 2014

Contents

1 Revision History

Revision #	Author	Date	Comment
1	Vitaliy Kondratiev, Nathan Johrendt, Tyler Lyn, Mark Gammie	October 28	Adding Test Plan Revision 0
2	Vitaliy Kondratiev, Nathan Johrendt, Tyler Lyn, Mark Gammie	October 29	Test Plan Updates
3	Vitaliy Kondratiev, Nathan Johrendt, Tyler Lyn, Mark Gammie	October 29	Test Plan Update
4	Vitaliy Kondratiev, Nathan Johrendt, Tyler Lyn, Mark Gammie	October 30	Final Update for Revision 0 Test Plan
5	Nathan Johrendt	February 3	Test Plan Update

2 Executive Summary

2.1 Introduction

This document is designed to outline testing methods and techniques that are to be used during and after development of the Porter Simulation. Below listed in detail are the main test factors and the rationale for choosing them. Following the main test factors is a comprehensive list of specific system tests including information about the Test's factor, life cycle phase, type, whether it is static or dynamic, manual or automated, and the specific techniques used to conduct the test.

2.2 Automation Tools

1. **Mock:** Mock is a Python library used for easy isolation of Python functions for testing and assertions
2. **Unittest:** Unittest is a Python framework for constructing automated tests
3. **Excel:** Excel will be used for analysing statistical information and verifying its correctness
4. **Virtual Box:** Virtual Box will be used when testing the portability of our software between different operating systems

2.3 Template

The template for this test plan document was derived from examining the testing section of the CodeClone example on avenue and through reviewing the test plan marking rubric to create a format that suits our simulation software.

3 Test Factors and Rationales

3.1 Reliability

Rationale: Since the simulation software is to be used by a non-technical staff, consistent execution and termination of the program is required. Individual simulations could run for long periods of time without requiring user interaction and are expected to terminate and store results without user supervision.

3.2 Ease of Use

Rationale: End users are non-technical so any interaction with the program, whether input or output, should contain a minimal amount of technical information.

3.3 Portability

Rationale: Control of the systems that the simulation will run on is left to the end users, so the implementation will be designed to function on a wide variety of industry-popular operating systems.

3.4 Correctness

Rationale: Since the simulation will be modelling real-world events, it needs to consistently generate accurate results that compare to recorded data provided by HHS.

4 Specific System Tests

4.1 Event List Correctness 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation event list is generated by initializing the simulation

(ii) **Input:** Event List of 50 events, manually constructed

(iii) **Description:** As the simulation runs, when an event is dispatched to a porter, it will then have fewer jobs left to assign

(iv) **Expected Output:** Conclude the event list contains 1 less job each time a porter is dispatched

4.2 Event List Correctness 2

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Static

Manual/Automated: Manual

Technique:

(i) **Initial State:** Event list is generated through initializing the simulation

(ii) **Input:** Event List of 5 events, manually constructed

(iii) **Description:** Examine the code by working through the process of popping off one event from the remaining list on paper

(iv) **Expected Output:** Conclude the event list contains 1 less job each time a porter is dispatched

4.3 Event List Correctness 3

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Automated

Technique:

(i) **Initial State:** Simulation event list of length 50 is generated by initializing the simulation

(ii) **Input:** Event List of 50 events, randomly constructed

(iii) **Description:** As the system is tested during the course of development a report is always generated after execution. This report allows each completed event to be checked for possible errors

(iv) **Expected Output:** System returns an output for every event that has come off the list

4.4 State Change Correctness 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation event list is generated by initializing the simulation, Porters states generated by initializing simulation and set to "Available"

(ii) **Input:** Single Event to be assigned to a Porter, Porter with state "Available"

(iii) **Description:** After single iteration of the simulation, execution of Event causes a single Porter to change state

(iv) **Expected Output:** Porter with state "Dispatched"

4.5 State Change Correctness 2

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Static

Manual/Automated: Manual

Technique:

(i) **Initial State:** Event list is initialized, all porters are initialized to "Available"

(ii) **Input:** A set of 5 "Pending" events, manually coded, and a set of 5 "Available" porters

(iii) **Description:** Examine the code manually by working through the process of single event being assigned to an "Available" Porter on paper

(iv) **Expected Output:** Conclude the Porter's state is changed to "Dispatched"

4.6 State Change Correctness 3

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Automatic

Technique:

(i) **Initial State:** Simulation event list of length 50 is randomly generated by initializing the simulation, 50 Porter states generated by initializing simulation and set to "Available"

(ii) **Input:** A set of 50 "Pending" events, randomly generated, and a set of 50 "Available" porters

(iii) **Description:** As the system is tested during the course of development a report is always generated after execution detailing the state of each Porter. This report can be examined to ensure that each event follows the pending, dispatched, in-progress and complete process.

(iv) **Expected Output:** System returns an output for every Porter state change

4.7 Porter/Event Linkage Correctness 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation event list is generated by initializing the simulation, Porters' states generated by initializing simulation

(ii) **Input:** A single "Pending" event, single "Available" Porter

(iii) **Description:** Every time an event moves from pending to dispatched, a porter must be linked to that event

(iv) **Expected Output:** Porter and Event are linked together uniquely by comparing to all other Porter/Event pairs

4.8 Porter/Event Linkage Correctness 2

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Static

Manual/Automated: Manual

Technique:

(i) **Initial State:** Assume simulation event list is generated by initializing the simulation, assume porters states generated by initializing simulation

(ii) **Input:** A single "Pending" event, Single "Available" Porter

(iii) **Description:** Examine the code and work through the process of linking a "Pending" event to a "Available" Porter on paper

(iv) **Expected Output:** Conclude that Porter and Event are linked together uniquely by comparing to all other Porter/Event pairs

4.9 Porter/Event Linkage Correctness 3

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Automated

Technique:

(i) **Initial State:** Simulation event list of length 50 is randomly generated by initializing the simulation, 50 Porter states generated by initializing simulation

(ii) **Input:** List of 50 "Pending" Events, randomly generated, set of "Available" Porters

(iii) **Description:** System records every linkage created during simulation and compares them to each other to ensure that events are dispatching properly

(iv) **Expected Output:** A report is generated detailing every link between Porter/Event. System compares all linkages and concludes they are unique.

4.10 Task Pool Correctness 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation event list is generated by initializing the simulation, porter states generated by initializing simulation all with states not equal to "Available"

(ii) **Input:** Single event, set of Porters with state not equal to "Available", empty task pool

(iii) **Description:** Events that cannot link to an "Available" Porter go to the task pool

(iv) **Expected Output:** Task pool contains one additional Event

4.11 Task Pool Correctness 2

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Static

Manual/Automated: Manual

Technique:

(i) **Initial State:** Assume simulation event list is initialized, assume porter states are initialized with all states not equal to "Available"

(ii) **Input:** Single event, Set of Porters with state not equal to "Available", Empty Task Pool

(iii) **Description:** Examine code and work through the process of events that cannot link to a "Available" Porter, and checking that those events go to the Task Pool on paper

(iv) **Expected Output:** Conclude that the Task Pool contains one additional event

4.12 TASK POOL CORRECTNESS 3

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Automatic

Technique:

(i) **Initial State:** Simulation event list of length 50 is randomly generated by initializing the simulation, 50 Porter states generated by initializing simulation

(ii) **Input:** List of 50 events, set of 50 porters

(iii) **Description:** Each time an event is added to the task pool the system generates a report

(iv) **Expected Output:** Report of all events that were added to the task pool

4.13 TERMINATION CORRECTNESS 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Task pool is empty of events and all porters are in state "Available"

(ii) **Input:** Empty Task Pool

(iii) **Description:** System recognizes initial state as a termination condition

(iv) **Expected Output:** Simulation termination

4.14 TERMINATION CORRECTNESS 2

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Static

Manual/Automated: Manual

Technique:

(i) **Initial State:** Assume Task pool is empty of events and assume all porters are in state "Available"

(ii) **Input:** Assume empty Task Pool

(iii) **Description:** Examine the code to conclude that a termination condition has been met

(iv) **Expected Output:** Conclude simulation termination

4.15 Termination Correctness 3

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Automated

Technique:

(i) **Initial State:** 50 simulations are initialized

(ii) **Input:** 50 simulations with termination conditions "Empty Task Pool, all Porters in state 'Available'"

(iii) **Description:** Testing system starts up 50 simulations and records which have terminated once reaching the termination condition

(iv) **Expected Output:** A report is generated by a testing system detailing what systems have terminated

4.16 Input/Initialization Correctness 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Uninitialized Simulation

(ii) **Input:** Simulation Input File (.csv extension), containing information; 10 Porters, 10 Events

(iii) **Description:** Once input file is loaded by the simulation, it has become initialized with modified values

(iv) **Expected Output:** Simulation initializes 10 Porters and 10 Events

4.17 Input/Initialization Correctness 2

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Automatic

Technique:

(i) **Initial State:** 50 uninitialized simulations
(ii) **Input:** 50 simulation input files (.csv extension) containing information; 1 to 10 Porters, 1 to 10 Events, 100 unique combinations
(iii) **Description:** Testing program takes in 50 input files and initializes 50 simulations, large number of results to cross-check for inconsistencies.
(iv) **Expected Output:** Testing program generates a report on all 50 simulation that were initialized

4.18 The Golden Test

Test Factor: Reliability & Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Uninitialized Simulation in a deterministic mode
(ii) **Input:** 'Gold Copy' set of events. 5 Events; Porter moving patient from Emergency to ICU using stretcher, Porter moving oxygen tank from storage to a room in Ward A, Porter moving specimen samples from laboratory A to laboratory B, Porter moving patient from X-Ray to Ultrasound using wheelchair, Porter takes a break
(iii) **Description:** Using a 'gold copy' of a set of events, new program builds are retested using the 'gold copy' to ensure consistency of execution by examining inconsistencies in the output file. This specific set of events is maintained and retested on new revisions of the software.
(iv) **Expected Output:** A difference file comparing 'gold copy' statistics against the newly tested code outlining the inconsistencies between them. The maximum variance allowed in the results is still being determined.

4.19 Usability Test 1

Test Factor: Ease of Use

Life Cycle Phase: Final Stages of Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation prior to execution
(ii) **Input:** End user with an example simulation set; Create Input file, Load Input File into Simulation, Initialize Simulation
(iii) **Description:** End user is provided with a set of instructions to Initialize the Simulation
(iv) **Expected Output:** The end user successfully Initializes the Simulation

4.20 Compatibility Test

Test Factor: Portability

Life Cycle Phase: Final Stages of Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Unknown Operating System with the simulation accessible on local storage

(ii) **Input:** 'Gold Copy' simulation parameters (details outline previously under 'The Golden Test')

(iii) **Description:** Simulation is run and compared to results computed on other operating systems to ensure it is functioning normally in the new environment

(iv) **Expected Output:** Simulated results are consistent with previously generated values

4.21 Distribution Correctness 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Structural

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** N/A

(ii) **Input:** Known distributions in our simulation: Poisson, geometric.

(iii) **Description:** Run the distribution when creating the event times and examine the results through a proven program that compares the regression (Excel or R). Determining real averages from provided data and closely comparing them to our generated statistical results will be essential to refining our software into an accurate simulation.

(iv) **Expected Output:** After examination the results follow the Poisson distribution

5 Proof of Concept Test

Proof of Concept Test will include these tests

(a) Event List Correctness 1

(b) State Change Correctness 1

(c) Task Pool Correctness 1

(d) Termination Correctness 1

(e) Input/Initialization Correctness 1

6 SCHEDULE

Phase #	Phase Name	Start Date	Description
1	Development Phase 1	November 1st	N/A
2	1st Proof of Concept	November 12th	20 Minute demonstration delivered for the rest of the capstone class
3	Development Phase 2	November 13th	N/A
4	2nd Proof of Concept	December 2nd	Demonstration delivered to the stakeholders
5	Development Phase 3	December 3rd	N/A
6	Test Phase 1	March 1st	N/A
7	Development Phase 4	March 15th	N/A
8	Testing Phase 2	March 28th	N/A
9	Final Demonstration	March 29th	N/A