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

March 17, 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
6	Nathan Johrendt	March 17	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

Use of these tools is being examined for both current and future testing

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. **pyunit:** A python unit testing framework
4. **Excel:** Excel will be used for analyzing statistical information and verifying its correctness
5. **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 both technical and 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 both technical and non-technical so any interaction with the program, whether input or output, should contain a minimal amount of complex 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 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 variables from interface, as well as a file location for both data.csv and schedule.csv for importing job data and porter shift information respectively.

(iii) **Description:** Once input values are loaded by the simulation, it has become initialized with modified values. Before beginning simulation, the simulation will re-print the information it just imported to confirm that it has found the right information.

(iv) **Expected Output:** Simulation outputs variable data identical to that defined by the interface.

4.2 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 through the interface

(ii) **Input:** Event list is generated from jobs stored in the data.csv file. The simulation will be configured to add jobs for one day, with 30 porters scheduled 24 hours a day.

(iii) **Description:** As the simulation runs, when an event is dispatched to a porter, it will then have fewer jobs left to assign. With 30 porters on a 24 hour schedule, no jobs will remain undone, and any that remain will represent an error to examine.

(iv) **Expected Output:** Conclude that no incomplete jobs remain

4.3 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 reading the simulation code

(ii) **Input:** Event list of 3 events, manually constructed from the data.csv file

(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 one less job each time a porter is dispatched

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 through the interface

(ii) **Input:** Event list is generated from jobs stored in the data.csv file. Porter schedule is generated from the schedule.csv file. The simulation will be configured to add jobs for one day, with five porters scheduled 24 hours a day.

(iii) **Description:** With only five Porters working, they will be required to change state frequently as they complete all of a regular day's tasks understaffed. Any problems, beyond the obvious delays, will result in a Porter getting stuck with a job at a particular state. This will be recorded in the output data and can be addressed.

(iv) **Expected Output:** All Porters complete jobs dispatched to them successfully

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 generated through reading the simulation code

(ii) **Input:** A set of 3 "Pending" events, manually extracted from data.csv , and a set of 3 "Available" porters

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

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

4.6 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 and porter schedule are generated by initializing the simulation through the interface

(ii) **Input:** Event list is generated from jobs stored in the data.csv file. Porter schedule is generated from the schedule.csv file. The simulation will be configured to add jobs for one day, with 30 porters scheduled 24 hours a day.

(iii) **Description:** Every time an Event moves from pending to dispatched, a Porter must be linked to that event. This test is to ensure that all pairings of job and Porter are unique, and that after completing a job, a porter will return to "available" and continue accepting new jobs.

(iv) **Expected Output:** Porter and Event are linked together uniquely

4.7 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 and porter schedule are generated by initializing the simulation through the interface

(ii) **Input:** Event list is generated from jobs stored in the data.csv file. Porter schedule is generated from the schedule.csv file. The simulation will be configured to add jobs for one day, with one Porter scheduled 24 hours a day.

(iii) **Description:** With only a single porter active, the majority of the jobs added by the simulation will simply be added to the dispatcher and wait. This test is to ensure that the dispatcher holds incomplete jobs correctly and that they are reported after the simulation completes.

(iv) **Expected Output:** Simulation output properly stores many incomplete job entries.

4.8 Task Pool Correctness 2

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Static

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation is initialized through reading the code

(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 are added to the Task Pool on paper

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

4.9 Task Pool Correctness 3

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Unit

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Initialized simulation

(ii) **Input:** dispatcherUnit.py unit testing file

(iii) **Description:** Through the use of UnitTest in python, this test checks that the dispatcher correctly adds a new job to the Task Pool if there are no available Porters. Three specific variables are tested for: Priority Weight, Appointment Factor and the length of the pending jobs list.

(iv) **Expected Output:** Conclude that the Task Pool contains one additional event as the number of pending jobs has increased by one.

4.10 Termination Correctness 1

Test Factor: Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation is initialized through the interface

(ii) **Input:** Event list is generated from jobs stored in the data.csv file. Porter schedule is generated from the schedule.csv file. The simulation will be configured to add jobs for five days, with 30 porters scheduled 24 hours a day.

(iii) **Description:** After accepting all inputs and initializing, the simulation will compute five days of operational data and output the results.

(iv) **Expected Output:** Simulation outputs that it has completed each step and has written the results to the dashboard.

4.11 The Golden Test

Test Factor: Reliability & Correctness

Life Cycle Phase: Throughout Development

Type: Functional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation is initialized through the interface

(ii) **Input:** 'Gold Copy' set of events imported from golddata.csv, as well as Porter shift information from goldschedule.csv

(iii) **Description:** Using a 'gold copy' of a set of events and Porter shifts, new program builds are retested using the 'gold copies' 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. A difference file can then be created in excel 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.

(iv) **Expected Output:** Simulation outputs that it has completed each step and has written the results to the dashboard.

4.12 Compatibility Test 1

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

4.13 Usability Test 1

Test Factor: Ease of Use

Life Cycle Phase: Final Stages of Development

Type: Nonfunctional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation prior to execution

(ii) **Input:** End user with a copy of both the simulation software package and accompanying user manual.

(iii) **Description:** End user is provided with a set of instructions from the user manual on how to initialize and run the simulation. The success of this test is determined by how much external assistance the End User requires from the development team on their first use of the software.

(iv) **Expected Output:** The end user successfully Initializes the Simulation

4.14 Performance Test 1

Test Factor: Ease of Use

Life Cycle Phase: Final Stages of Development

Type: Nonfunctional

Dynamic/Static: Dynamic

Manual/Automated: Manual

Technique:

(i) **Initial State:** Simulation prior to execution

(ii) **Input:** Any combination of inputs that the interface will accept and begin simulating

(iii) **Description:** The project stakeholders have placed a loose time limit on the duration of execution for a single simulation run of fifteen minutes. No correct simulation execution has taken longer than five minutes, with most between one and two minutes, but ensuring this general timing on each system the simulation is run on is the purpose of this test.

(iv) **Expected Output:** The simulation completes execution taking between thirty seconds and five minutes

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

Note: Not reflective of our current schedule.

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