

MCMASTER UNIVERSITY

CAS 4ZP6

TEAM 9

CAPSTONE PROJECT 2013/2014

PORTER SIMULATION

Design Revision 0

Authors:

Vitaliy Kondratiev - 0945220

Nathan Johrendt - 0950519

Tyler Lyn - 0948978

Mark Gammie - 0964156

Supervisor:

Dr. Douglas Down

January 14, 2014

CONTENTS

1	Revision History	3
2	Executive Summary	3
2.1	Introduction	3
2.2	Purpose	3
2.3	Design Overview	3
3	Implementation Material	3
3.1	Language of Implementation	3
3.2	Supporting Technology and Frameworks	3
4	Process Diagram	4
5	Dependency Diagram	6
6	Decomposition Description	7
6.1	Core - Simulation Core	7
6.2	Core - Simulation State	7
6.3	Core - Task	8
6.4	Core - Task List Builder	9
6.5	Core - Porter	10
6.6	Core - Dispatcher	11
6.7	Import - Simulation Setting	12
6.8	Import - Hospital Layout Graph	12
6.9	Import - Statistical Data Import	13
6.10	Export - Statistical Data Export	13
6.11	Logging	14
6.12	GUI - Basic Settings	14
7	Anticipated Changes	17

1 REVISION HISTORY

Revision #	Author	Date	Comment
1	Vitaliy Kondratiev, Nathan Johrendt, Tyler Lyn, Mark Gammie	January 11, 2014	Revision 0 Added to repository

2 EXECUTIVE SUMMARY

2.1 INTRODUCTION

This document outlines the design decisions, style and methodology for the project of Porter Simulation to be complete for Hamilton Health Sciences.

2.2 PURPOSE

The purpose of this document is to outline the design of each component and how they interface between each other. This document will aid the developers in the development process as well as any future maintenance required.

2.3 DESIGN OVERVIEW

3 IMPLEMENTATION MATERIAL

3.1 LANGUAGE OF IMPLEMENTATION

Visual Basic in Excel will be used to interface GUI to Simulation Core module. Python Version 2.7.5 will be used for Simulation Core module and all other modules.

3.2 SUPPORTING TECHNOLOGY AND FRAMEWORKS

Simulation will be built on the SimPy 3.0.2 library <https://simpy.readthedocs.org/en/latest/>. GUI will be built in Excel 2010.

4 PROCESS DIAGRAM

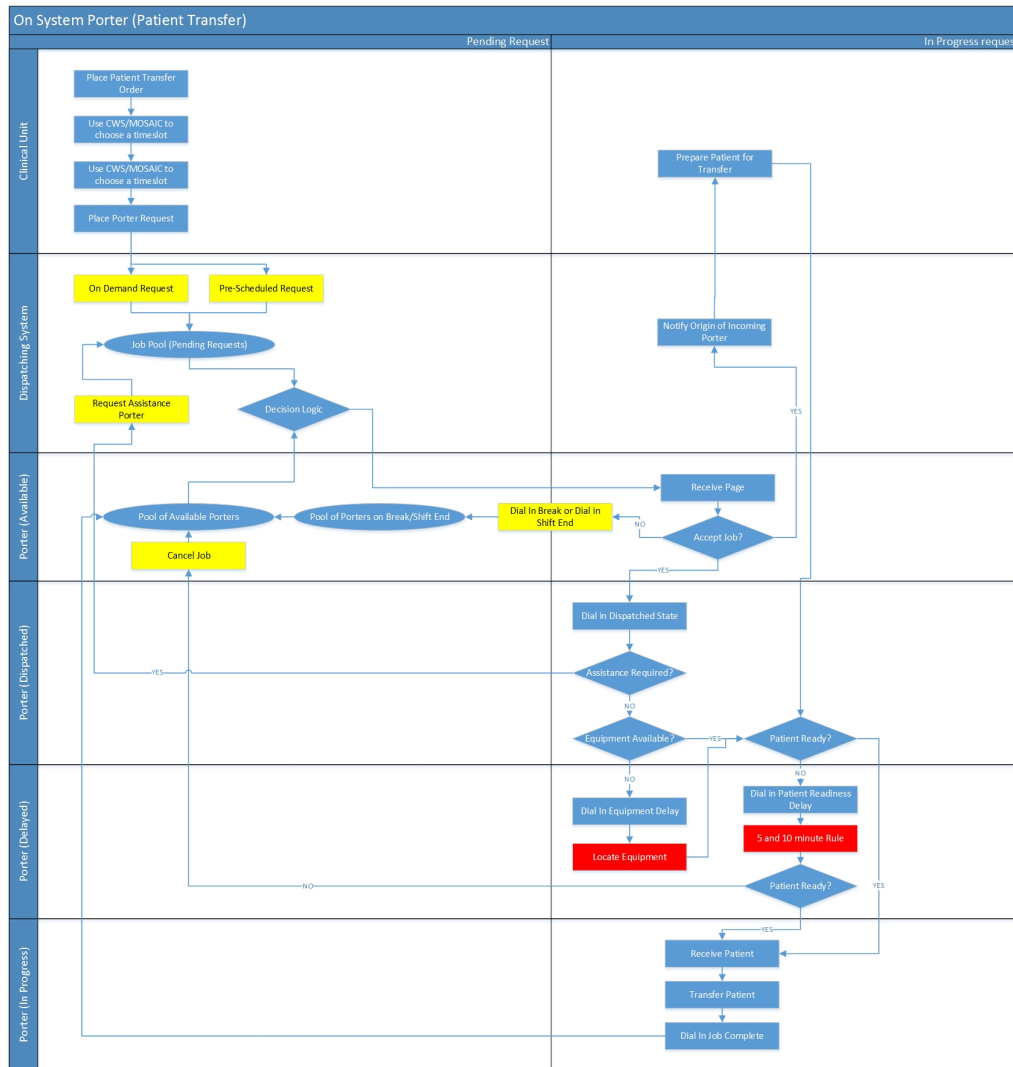


Figure 4.1: Process Diagram

5 DEPENDENCY DIAGRAM

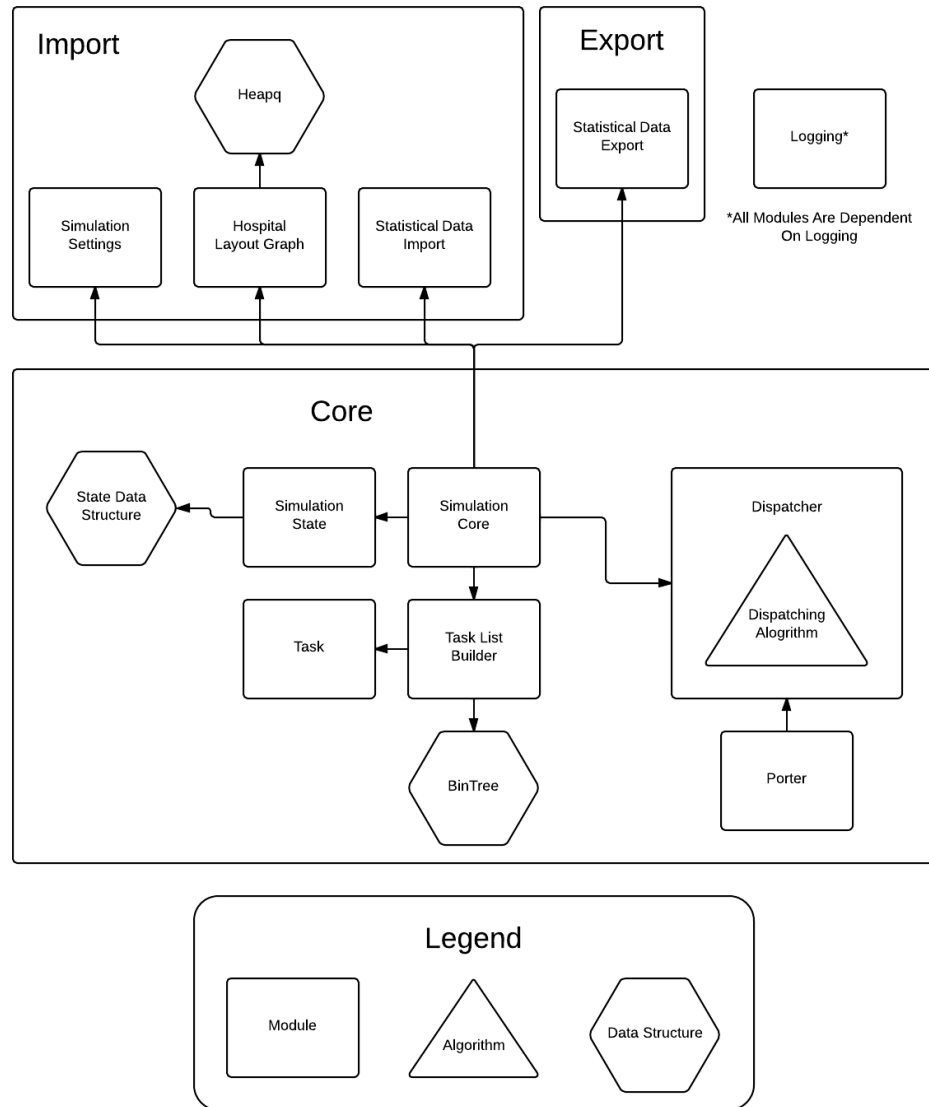


Figure 5.1: Dependency Diagram

6 DECOMPOSITION DESCRIPTION

6.1 CORE - SIMULATION CORE

Type: Module

Purpose: This module calls the required functions to fetch the import data, initializes the simulation and initializes processes.

Function: This module calls the required import modules (Simulation Settings, Hospital Layout Graph, Statistical Data Import), passes their data so that the Simulation State, Event List Builder and Dispatcher can be initialized.

Interface:

The interface for the Simulation core is the command line used to called it, defined as follows:

SimCore settings_filename graph_filename data_filename

settings_filename: the name of the file containing the simulation settings

graph_filename: the name of the file containing the hospital layout graph

data_filename: the name of the file containing the statistical data

Process Steps: This module first uses the Simulation Settings, Hospital Layout Graph and Statistical data to read in the external data. This data is used to configure other core modules in the simulation.

Once the external data is imported and the core modules are configured, the Simulation Core will begin the simulation and manage the core modules.

Data: None

Error Handling: Catch all on the simulation loop to report pertinent errors and to prevent unexpected termination.

Requirement Reference: 11.5

Critical Revision 0 Component: True

6.2 CORE - SIMULATION STATE

Type: Module

Purpose: This module's purpose is to be an interface between the simulation and its simulation state data structure.

Function: This module will contain functions that will be used by the simulation to perform queries on the system state data structure.

Interface:

initSimulateState():

- Instantiating the simulation state with null values

getSimulationTime():

- Returns the current simulation time

getPorterList():

- Returns a list of the porter objects

Process Steps:**Data:****Error Handling:****Requirement Reference:**

Critical Revision 0 Component: True

6.3 CORE - TASK

Type: Module

Purpose: Provide information about a job to the porter.

Function: The task is declared with an origin, destination, required equipment and a priority. This information is necessary to allowing a porter to complete a job.

Interface:

getOriginLocation():

- Returns the origin of a job

getDestinationLocation():

- Returns the destination of a job

getEquipment():

- Returns the equipment required for a job

getPriority():

- Returns the priority of a job

getTime():

- Returns the time that the task will be requested

updatePriority():

- A process that will set the job to a higher priority after a specified amount of time has passed.

Process Steps: Tasks will be initialized once the Simulation Core shares the import modules with the Event List Builder. An initialized task will be able to provide the Dispatcher all the required information to organize a list of tasks. Also, once a task is given to a porter, the porter will then be able to start and complete a transport job.

Data: Transport Job Information

Error Handling: Catch any null values returned by the task's properties

Requirement Reference: None

Critical Revision 0 Component: True

6.4 CORE - TASK LIST BUILDER

Type: Module

Purpose: Produces the list of tasks for the Simulation Core and Dispatcher modules to process

Function: This module populates a series of tasks following either a distribution or statistical data.

Interface:

addTask(origin, destination, equipment, priority, time):

- Input the origin, destination, equipment and priority of a job. Also include the time the task will be added to the Dispatcher

setTaskBuilderStats(data)

- Input the statistical or distribution data as it is required for creating meaningful data

Process Steps: The Task List Builder will first receive the data from the Simulation Core. This data will allow the Task List Builder to create jobs based on user specified parameters. Secondly, the Task List Builder will generate tasks and append them to the BinTree data structure, to allow for ordering, inserting and retrieval of tasks. Once a task reaches its "time" it will be added to the Dispatcher where it can be given to an available porter.

Data: Task List BinTree

Error Handling: Catch any null tasks

Requirement Reference: 8.1 (d)

Critical Revision 0 Component: True

6.5 CORE - PORTER

Type: Module

Purpose: To complete jobs provided by the dispatcher

Function: Completes the transport jobs assigned by the dispatcher. Unless a job is cancelled the porter will traverse through four states ('pending', 'dispatched', 'inprogress', 'complete')

Interface:

setStatePending(state):

- Input the pending state
- Sets the porter's state to pending and waits to be assigned a job

setStateDispatched(state):

- Input the dispatched state
- Sets the porter's state to dispatched and calculates the time between the porter's location and the job's origin

setStateInprogress(state):

- Input the inprogress state
- Sets the porter's state to inprogress and calculates the time between the job's origin and destination

setStateComplete(state):

- Input the complete state
- Sets the porter's state to complete, records the completion time and sets the porter back to the pending state

getAutoLocation():

- Returns the estimated location of a pending porter
- Estimates the current location of a porter based on how many minutes they have been in
- Output the estimated location of a pending porter
- Estimates the current location of a porter based on how many minutes they have been in the pending state.

Process Steps: The module listens for state changes provided by the dispatcher and updates its' internal components as necessary.

Data: Stores internal data relating to its' current state.

Error Handling: Not Available

Requirement Reference: Not Available

Critical Revision 0 Component: True

6.6 CORE - DISPATCHER

Type: Module

Purpose: To organize pending jobs based on a weighted-value and assign them to porters

Function: This module orders pending jobs based off of a Dispatch Value which is computed using several parameters (Proximity Match Value, Weighted Job Priority and Appointment Factor). The pending job with the greatest Dispatch Value will be assigned to the closest available porter. Once the job is assigned to the porter the job will be considered as a dispatched job.

Interface:

assignJob(Task):

- Assigns the job with the greatest Dispatch Value to the closest available porter.

getProximityMatchValue(TaskOrigin):

- Input the origin of a pending job
- Output a value based on how close an available porter is to a job's origin

getWeightedJobPriority(TaskOrigin, TaskDestination):

- Input the origin and destination of a pending job
- Output a value based on the priority of the pending job

getAppointmentFactor(Task):

- Input a pending job
- Update the value for a job depending on if it was pre-scheduled or on-demand.

getDispatchValue(Task):

- Input a pending job
- Compute the DispatchValue for a job: $(\text{ProximityMatchValue} + \text{WeightedJobPriority} * \text{AppointmentFactor})$

Process Steps: All pending jobs are assessed and given a dispatch value (DV) based on the weighting and values of specified dispatch parameters.

These weights and values are determined using either the location of an available porter or the priority of a pending job.

All of the pending jobs are then ordered from greatest dispatch value to the least. When there is an available porter the pending job with the greatest dispatch value is given to the closest porter.

Data:

- Pending jobs

Error Handling: Not Available**Requirement Reference:** Not Available**Critical Revision 0 Component:** True

6.7 IMPORT - SIMULATION SETTING

Type: Module**Purpose:** Receives the settings data from the GUI and translates them for use by the simulation core**Function:** Receives data in the form of a csv and converts it to THE FORMAT file**Interface:**

translateSettings(Settings):

- Input, the csv settings file
- Output, translated settings

Process Steps:**Data:****Error Handling:****Requirement Reference:****Critical Revision 0 Component:** True

6.8 IMPORT - HOSPITAL LAYOUT GRAPH

Type: Module**Purpose:****Function:****Interface:****Process Steps:****Data:**

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

6.9 IMPORT - STATISTICAL DATA IMPORT

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

6.10 EXPORT - STATISTICAL DATA EXPORT

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

6.11 LOGGING

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

6.12 GUI - BASIC SETTINGS

Type: User Interface

Purpose: Allows the user to change the basic setting of the simulation and run the simulation

Function:

Number of Porters: specify the number of porters for the simulation to run

Start Date: specify the day the simulation will run from

Start Time: specific time the simulation runs from

End Date: specific time day the simulation will end

End Time: specific time the simulation ends

Job Distribution: users can choose from predefined distributions or base it on existing statistical data

Job Intensity: specify the frequency of job distribution

Correct Equipment Usage: specify the percentage of correct equipment events

Patient Readiness: specify the percentage of ready patients on porter arrival

Porter Wait Time: specify the time a porter waits for patient to be ready before abandoning job

Interface:

Advanced Setting: proceed to Advanced Setting GUI

Simulate: push settings to Simulation Core

Default Settings: reset to default values

Process Steps: Not Available

Data: Not Available

Error Handling: If any of the below values violates its restriction, excel will not allow the information to be sent to the simulation core.

Number of Porters: restricts the number of porters to a positive integer

Start Date: restricts the start date to day/month/year format

Start Time: restricts the start time to 12 or 24 hour time

End Date: restricts the end date to day/month/year format and checks that the date is on the same date or a later date than the start date

End Time: restricts the end time to 12 or 24 hour time and checks that the end time is further in the future than the start time

Job Distribution: user is restricted to a set series of options

Job Intensity: user is restricted to a set series of options

Correct Equipment Usage: restricts the value between 0 and 100 percent

Patient Readiness: restricts the value between 0 and 100 percent


Porter Wait Time: restricts the value to a minimum of 0

Requirement Reference: 10.1.1, 10.1.2, 10.1.3, 10.1.4

Critical Revision 0 Component: True

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER LOAD TI

J11 : ✕ ✓ fx min: 0% max: 100%

	A	B	C	D	E	F	G	H	I	J	K	L			
1						Porter Simulation									
2						Basic Setting				Hint					
3						Number of Porters		50		▲▼		min: 0 max: 200			
4						Start Date		14-Apr-14		▲▼		must be less than or equal to			
5						Start Time		7:30 AM		▲▼		must be less than or equal to			
6						End Date		15-Apr-14		▲▼		must be more than or equal to			
7						End Time		3:30 PM		▲▼		must be more than or equal to			
8						Job Distribution		Data Based		▲▼		Choose from Available List			
9						Job Intensity		Moderate		▲▼		Choose from Available List			
10											Compliance				
11	Correct Equipment Usage		80%		▲▼		min: 0% max: 100%								
12	Patient Readiness		30%		▲▼		min: 0% max: 100%								
13	Porter Wait Time		5 minutes		▲▼		min: 0 max: 60 min								
14						Advanced Setting									
15						Simulate									
16															
17															
18															
19															

Main (+)

READY

Figure 6.1: Basic GUI

7 ANTICIPATED CHANGES

1 Change 1