McMaster University

CAS 4ZP6

Team 9

Capstone Project 2013/2014

Porter Simulation

Design Revision 0

Authors:
Vitaliy Kondratiev
Nathan Johrendt
Tyler Lyn
Mark Gammie

Supervisor:
Dr. Douglas Down

January 13, 2014

CONTENTS

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

2.2 Purpose

2.3 Design Overview

3 Implementation Material

3.1 Language of Implementation

3.2 Supporting Technology and Frameworks

3.3 PROCESS DIAGRAM

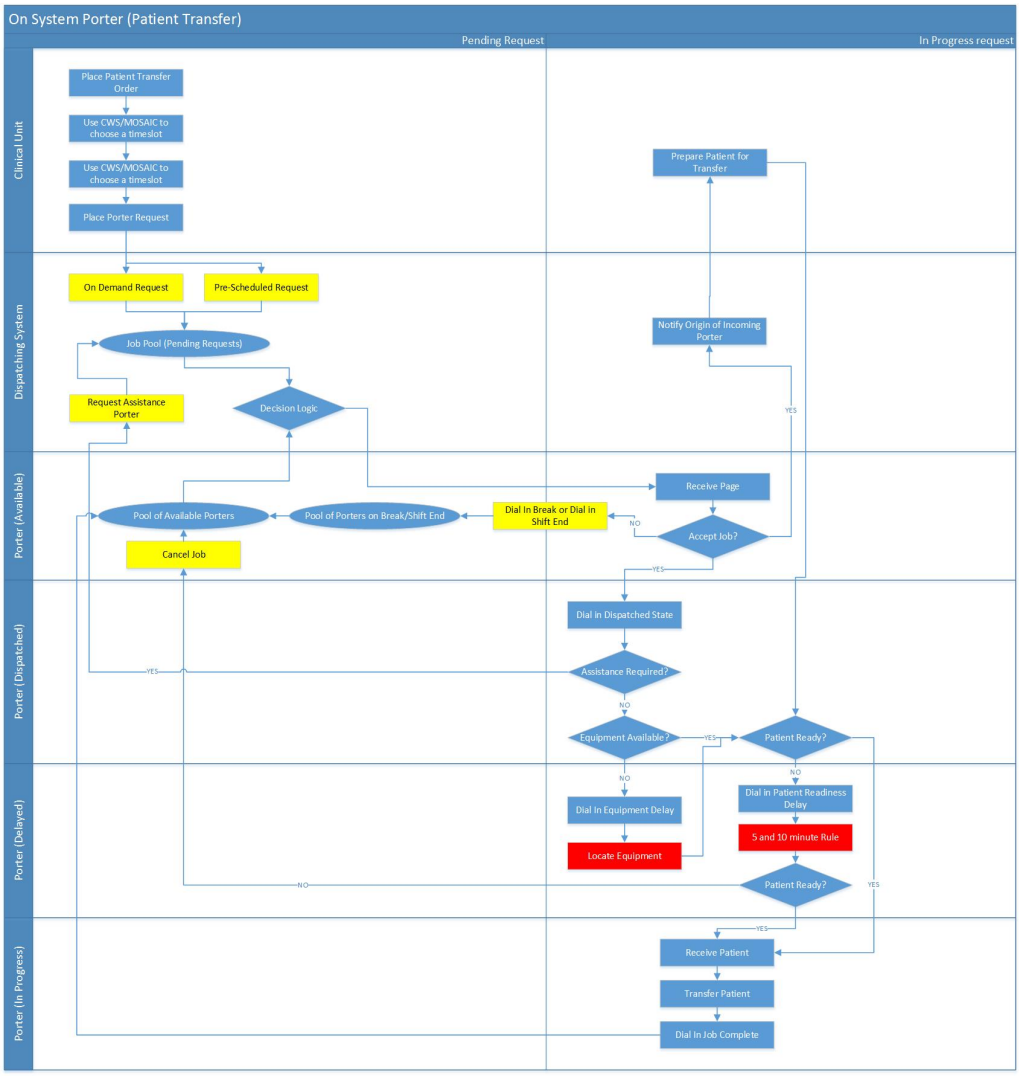

Figure 3.1: Process Diagram

4 Dependency Diagram

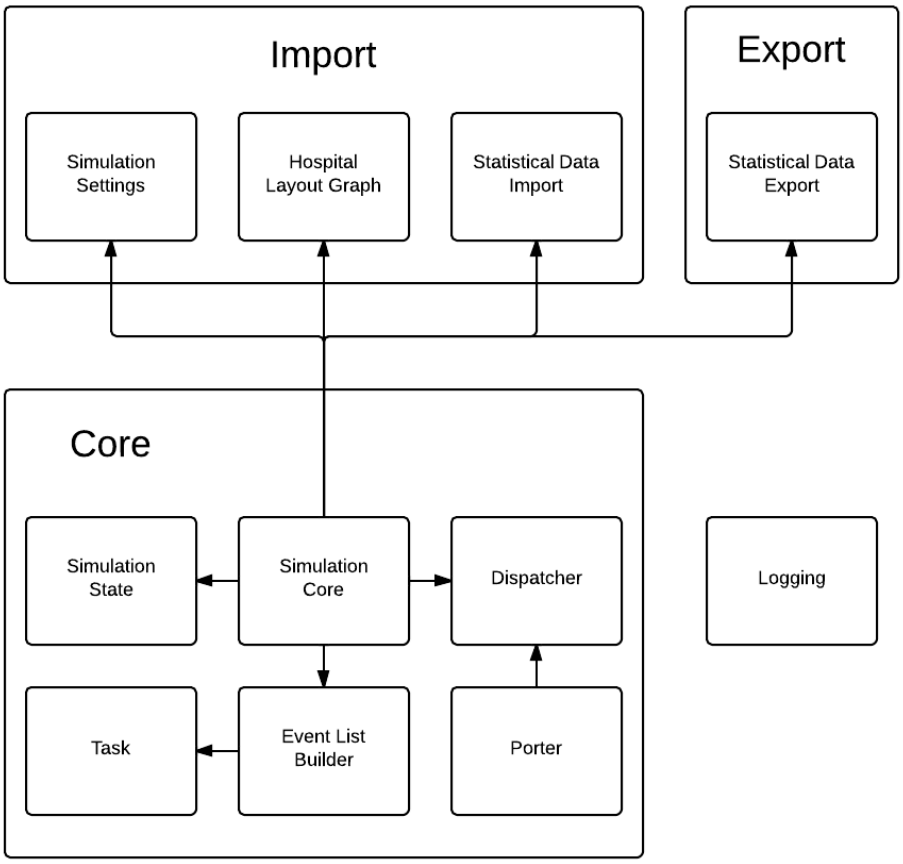

Figure 4.1: Dependency Diagram

5 Decomposition Description

5.1 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

5.2 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

5.3 CORE - TASK

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

5.4 CORE - EVENT LIST BUILDER

Type: Module

Purpose: Produces the list of events for the Simulation Core module to process

Function: Takes Task List as input to produce Event List

Interface: nextEvent(): upon query takes the top event from the stack and returns it

Process Steps: TBD

Data: Built on BinTree

Error Handling: TBD

Requirement Reference: 8.1 (c)

Critical Revision 0 Component: True

5.5 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():

- 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

5.6 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(Job):

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

getProxmityMatchValue(Job Origin):

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

getWeightedJobPriority(Job Origin, Job Destination):

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

getAppointmentFactor(Job):

- Input a pending job

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

getDispatchValue(Job):

- Input a pending job
- Compute the DispatchValue for a job: (ProxmityMatchValue + WeightedJobPriority * AppointmentFactor)

updateJobPriority(Job):

- Input a pending job
- Determine if the pending job has been waiting too long. If the job has been pending for a specified amount of time, update it to a higher priority.

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

5.7 IMPORT - SIMULATION SETTING

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

5.8 IMPORT - HOSPITAL LAYOUT GRAPH

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

5.9 IMPORT - STATISTICAL DATA IMPORT

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

5.10 EXPORT - STATISTICAL DATA EXPORT

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

5.11 LOGGING

Type: Module

Purpose:

Function:

Interface:

Process Steps:

Data:

Error Handling:

Requirement Reference:

Critical Revision 0 Component: True

6 ANTICIPATED CHANGES

1 Change 1