SRS

For

Global Package Courier Tracking

COMP 4081 Software Engineering
(10 October 2014)

BitRunners (Team #5)

Ismael Alonso, Dereje Arega, Chris Hubbard, Ashlesh Gawande, Matthew Longley

Stephen Moo-Young

Version 1.2

Abstract

In this document, the specifications and requirements of our product, as well as its general characteristics, will be discussed. The purpose of our product is simulating the operations of a cargo airport. Our goal is to deliver significant and reliable statistics concerning the airport to our customers. As will be further explained in the document, our product will consist of two parts, a web interface and a compiled simulator. Due to the nature of these two parts, the main constraint we will experience through the development of our product is successfully establishing communication amongst them. Although the project is not big, we decided to use an incremental model, since we may need to change some characteristics on the go. The components of the team and their roles are the following: Ismael Alonso, project leader; Dereje Arega, software engineer; Ashlesh Gawande, lead programmer; Chris Hubbard, requirements engineer; and Stephen Moo-Young, test engineer. These are all managing roles; the manager of an area will divide and assign the tasks to be done regarding his area among the team members.

Table of Contents

Table of Figure(s)	
Figure 1	***Context diagram***
Figure 2	***Use case***
Figure 3	***DFD (Level 0)***

List of Tables

1. Introduction

The SRS is a document that details the requirements specified by the customer/user. Here in we describe the purpose, scope and overview of the SRS by our project.

1.1. Purpose

The purpose of the SRS is to describe our program and how it will function by listing functional and non-functional requirements for our software program development. This document will create a bridge between our team and our consumers and convene an understanding of the software being developed. This document is intended for users and developers of our project.

1.2. Scope

The name of our software product is Airport Process Time Simulator. The program simulates FedEx airport activity, such as de-birthing/birthing, storm cycles, and runway queue. Customers using this program have the freedom to choose their own parameters. The data provided by the simulation can then be analyzed to model a more efficient cargo airport. Other benefits include accurate projections for real-life modeling and simulation with extreme constraints.

1.3. Definitions, acronyms, and abbreviations

- SRS: Software Requirements Specifications
- GUI: Graphical User Interface
- XML: eXtended Markup Language

1.4. References

"Diagrams Done Right." *Flow Chart Maker & Online Diagram Software*. N.p., n.d. Web. 12 Sept. 2014.

IEEE Recommended Practice for Software Requirements Specifications. New York, NY: Institute of Electrical and Electronics Engineers, 1998. Web. 25 Sept. 2014.

1.5. Overview

Chapter 2 of this document, the Overall Description, gives a concise overview of our program. Each section in Chapter 2 further expands this in detail. Chapter 3, Specific Requirements, details the requirements specifications that this program will entail.

2. Overall Description

This chapter will describe everything necessary for the user to understand and use the product.

2.1. Product perspective

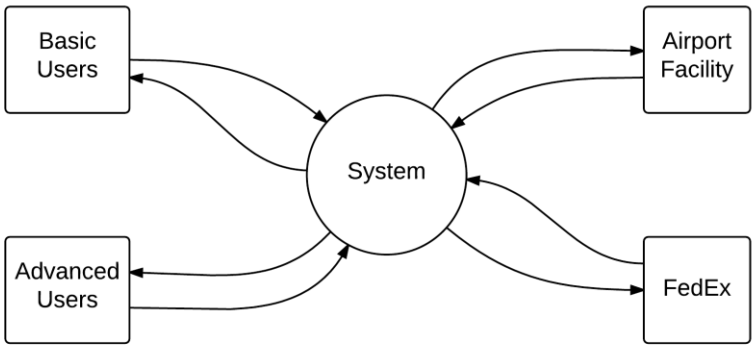

Figure 1: Context Diagram

The system will be used by two types of user advanced and basic. The company (FedEx) is interested in this problem as they can optimize their operations for maximum efficiency. The airport is the facility where this system's results would be applied and tested.

2.2. Product functions

- The purpose of this program is to run an accurate simulation for FedEx airport activity. The basic user will run the simulation without default values already specified, but advanced user should be able to change the following:
- The software shall require a GUI will provide the user an interface where parameters can be provided, the simulation can be started, and the data from the simulation shall be delayed.
- There shall also be a run function that will take all given parameters from the user in the GUI and relay it to the simulation.
- The software shall also contain a function to run the simulation itself successfully, with the given (or default) parameters, and gather the data specified in section 3.
- Once this is done the product shall display the results and also allow the user to run more simulations without restarting the software.

2.3. User characteristics

The user of this program acts as the consumer who specified the requirements of the simulation given in section 3. Therefore the user is expected to understand the results presented, what the parameters mean, and also reasonable values for said parameters. However we want to make the program intuitive to use. The GUI will give default values for every field in the form, so that anyone will be able to use the program. Figure 2 describes different user levels and how they can

change the default parameters to run the simulation.

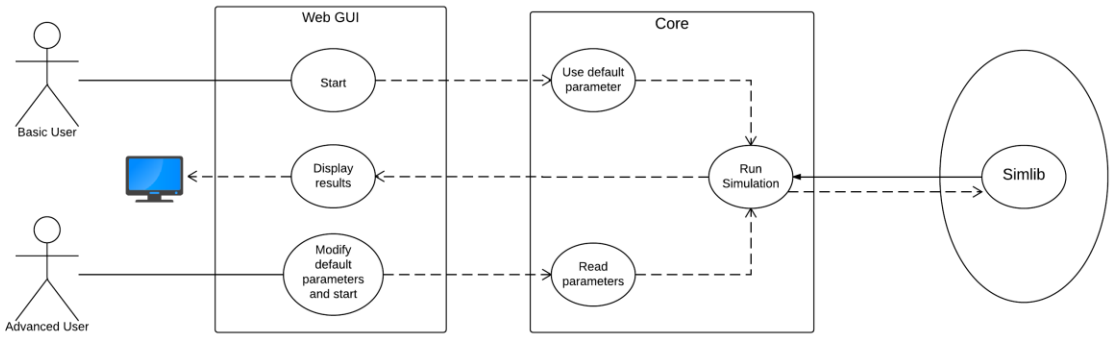

Figure 2: Use case

2.4. Constraints

- Communication: the existence of different parts using varying systems and languages might constrain the communication between these parts. The computational engine will be written in C/C++, while the GUI will be written in HTML/CSS with JavaScript. We cannot pass values as parameters straight from one to the other. Another communication system must be devised.
- Simlib: We do not know have a full understanding of this software yet.

2.5. Assumptions and dependencies

- JavaScript being able to read/write files in the local machine
- The user can operate a browser

If results are within constraints it is assumed that this simulation is an accurate representation of FedEx courier tracking.

3. Specific Requirements

This section contains all software requirements necessary for our project design.

3.1 Functional Requirements

A1 GUI Data Processor

The program shall have a module to handle the data input by user.

A1.1 Receive Input

The program shall receive input from user

- Purpose: Build a string one piece at a time that will eventually be written into a file
- Inputs: Part of the string that is already built, the ID of a field in the form, and a suffix
- Output: None
- Operations: concatenates the input string with the content of a field and the suffix

A1.2 Validate Input

The program shall validate the input it receives

- Purpose: Validate input that the user entered
- Inputs: Parameters in input field
- Output: In case of wrong input display error
- Operations: Will check the GUI input

A1.3 Build Input String

The program shall build an input string

- Purpose: Create a string containing the inputs in the specified format
- Inputs: User inputs from the GUI
- Output: A string
- Operations: Concatenation of inputs into a single string.

A1.4 Write Input File

The program shall write the input

- Purpose: communication between the GUI and the computational engine
- Inputs: The string to be written, file path
- Output: files in hard drive
- Operations: write the input string into the file

A1.5 Read Results

The program shall read the output

- Purpose: to read the file created by the computational engine
- Inputs: file path
- Output: string containing the XML
- Operations: display output

A1.6 Parse Results String

The program shall parse the output string

- Purpose: To put the results into a data structure
- Inputs: The result string
- Output: The data structure containing the results
- Operations: parsing XML

A1.7 Display Results

The program shall display the output

- Purpose: show the user the results of the simulation
- Inputs: the results that the parser provides from processing the output string
- Output: the results displayed to the screen

- Operations: linking the correct results with the GUI elements

A2 Computational Engine

Computation engine is core of the project which will work on around of Simlib to compute the simulation.

A2.1 Read Input File

The program shall read the input

- Purpose: Load the input file
- Inputs: String containing file path
- Outputs: String within the file
- Operations: reads the file

A2.2 Process Input

The program shall process the input

- Purpose: transform the input file into data that is manageable by the computational engine
- Inputs: the string from the file
- Output: data structure containing the information in the input
- Operations: reads the content of a string and writes into a data structure

A2.3 Simlib Process

- Purpose: Run the simulation
- Inputs: Structured input data
- Output: Raw simulation data
- Operations: TBD

A2.4 Calculate Results

The program shall calculate the output

- Purpose : to provide a readable and understandable result
- Inputs: State of different elements through the simulation
- Output: time statistics
- Operations: Calculate mean

A2.5 Build Output

The program shall build the output

- Purpose: creating a string in XML format that includes all the results from our simulation, one piece at a time
- Inputs: part of the string that is already built, indentation state, result of the simulation for a specific element
- Output: a string
- Operations: concatenates the string with the indentation state and the result of the simulation following the XML format that we specify for the file

A2.6 Write Results File

The program shall write the output

- Purpose: communication between the computational engine and the GUI
- Inputs: the string created by the build output, file path
- Output: file to the hard drive
- Operations: writes the file

3.2 Non Functional Requirements

- The program shall be efficient

- The program shall be easy to use
- The program shall be reliable
- The program shall be secure
- The program shall be safe

3.3 User interfaces

- Web elements: buttons, radio buttons, textbox

3.4 Software interfaces

- Input File
- 2 Output files- one that stores data, one that indicates data is ready

3.5 System features

Following are the system features

3.5.1 System Feature 1

Graphic for taxi information from simulation

3.5.1.1 Introduction/Purpose of feature

Inform user about the various taxi info

3.5.1.2 Stimulus/Response sequence

Output from the simulation

3.5.2 System feature 2

Graphic for berth information from the simulation

3.5.2.1 Introduction/Purpose of feature

Informs user about various berth information

3.5.2.2 Stimulus/Response sequence

Output from the simulation

3.5.3 System Feature 3

Graphic for queue information

3.5.3.1 Introduction/Purpose of feature

Inform user about various queue time averages

3.5.3.2 Stimulus/Response sequence

Output from the simulation

3.5.4 System Feature 4

Graphic for port info

3.5.4.1 Introduction/Purpose of feature

Inform user about the port residence time

3.5.4.2 Stimulus/Response sequence

Output from the simulation

3.6 Performance requirements

Need browser that can handle JavaScript

3.7 Design constraints

The simulation must finish in an appropriate amount of time.

The GUI must be intuitive

3.8 Software system attributes

Uses C++

Appendix A

- Requirements Traceability Matrix (RTM)

Req. ID System Level.	Req. ID Sub-system level	DFD Identifiers(s)	Module Names(s)	Verification Method	Tested
A1		1	GUI Data Processor		
	A1.1	1.1	Receive Input	T/D	
	A1.2	1.2	Validate Input	T/D	
	A1.3	1.3	Build Input String	T/D, I	
	A1.4	1.4	Write Input File	I	
	A1.5	1.5	Read Results	T/D, I	
	A1.6	1.6	Parse Results String	T/D	
	A1.7	1.7	Display Results	T/D	
A2		2	Computational Engine		
	A2.1	2.1	Read Input File	T/D	
	A2.2	2.2	Process Input	A	
	A2.3	2.3	Simlib process	A	
	A2.4	2.4	Calculate Results	A	
	A2.5	2.5	Build Output String	I, T/D	
	A2.6	2.6	Write Results File	I	

Table 1 Requirements Traceability Matrix

Appendix B

- DFD (Level 0)

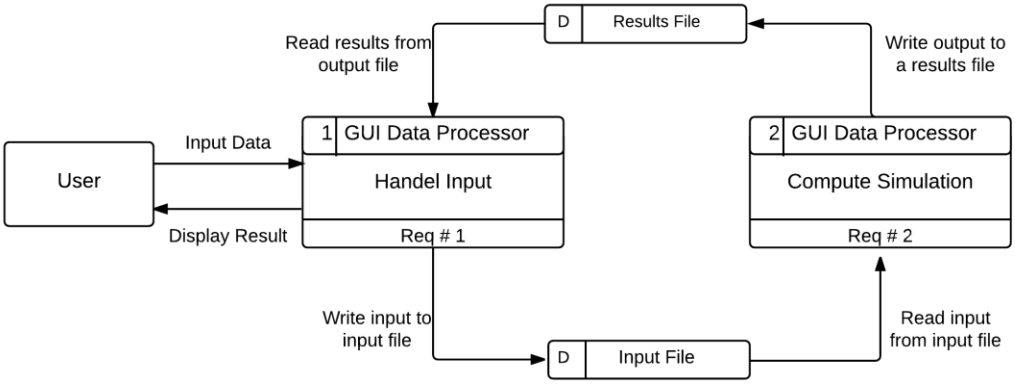

Figure 3 Data Flow Diagram (Level 0)