

Design Notebook

For

Global Package Courier Tracking

COMP 4081 Software Engineering
(26 October 2014)

BitRunners (Team #5)

Ismael Alonso, Dereje Arega, Chris Hubbard, Ashlesh Gawande, Matthew Longley
Stephen Moo-Young

Version 1.0

Abstract

In this document, our methodology as well as our design will be outlined in detail. 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

1	<i>Introduction.....</i>	<i>1</i>
1.1	Project Purpose and Goals	1
1.2	Design Approach	1
1.3	Traceability Approach	2
1.4	Background	2
1.5	Document Organization	2
2	<i>Requirements Analysis.....</i>	<i>3</i>
2.1	Graphical User Interface	3
2.2	Software Interface	3
2.3	Functional Requirements	3
2.3.1	GUI Data Processor	3
2.3.2	Computational Engine	3
2.4	Performance Requirements	4
2.5	Design Constraints	4
2.5.1	Utilize the Simlib Library	3
2.5.2	Delete and Insert Functions	3
2.5.3	Provided Parameters for Simulation	3
3	<i>Design</i>	<i>5</i>
3.1	Context Diagram	5
3.2	Level 0 Data Flow Diagram and P-specs.....	5
3.3	Level 1/2 Data Flow Diagram and P-specs.....	7
3.4	Transaction Analysis.....	10
3.5	Transform Analysis and Structure Chart	12
4	<i>References</i>	<i>13</i>
5	<i>Glossary</i>	<i>13</i>
	<i>APPENDIX A: GUI Prototype</i>	<i>14</i>
	<i>APPENDIX B: Project Schedule</i>	<i>15</i>
	<i>APPENDIX C: Requirements Traceability Matrix</i>	<i>17</i>
	<i>APPENDIX D: Identified Test Cases</i>	<i>18</i>

<i>Table of Figure(s)</i>	
<i>Figure 1</i>	<i>Context diagram</i>
<i>Figure 2</i>	<i>DFD (Level 0)</i>
<i>Figure 3</i>	<i>Level 1(1.n) DFD</i>
<i>Figure 4</i>	<i>Level 1(2.n) DFD</i>
<i>Figure 5</i>	<i>Level 2 (2.3.n) DFD</i>
<i>Figure 6</i>	<i>System DFD Partition</i>
<i>Figure 7</i>	<i>APTS Chart</i>
<i>Figure 8</i>	<i>GUI Prototype</i>

List of Tables

Table 1: Project Schedule

Table 2: Requirements Traceability Matrix

1. Introduction

This notebook will outline the design for Airport Process Time Simulator (APTS). Here in we describe the purpose, scope and overview of the Design Notebook by our project.

1.1. Project Purpose and Goals

The purpose of the APTS is to simulate 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. Our goal is to deliver a functional, intuitive software program that will realize our purpose above and give a realistic interpretation of airport courier tracking.

1.2. Design Approach

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.

We are using SSA/SD Structured System Analysis and the following steps:

Steps 1-2: Structured systems analysis

Step 3: The transaction analysis

Step 4: Identify the central transform in the DFD

Step 5: Merge the Structure Charts

Traceability Approach

In our Software Requirements Specification document we create our functional and non-functional requirements and put them into our requirements traceability matrix. We then designated a testing method to verify that each requirement would be satisfied. Finally we created various Design Flow Diagrams to show the correlation between our RTM and each process.

Background

Airport Process Time Simulator (APTS) is a program that uses Simlib as a backbone to run the simulation. This will interface with the GUI which the user will access from a webapp.

Organization of this document

Section 1 of this document provides the introduction to our project software and our purpose and goals. Section 2 breaks down our functional and non-functional requirements. Section 3 gives an overview of our design, shows our context diagram and analyzes each level of our DFD. Section 4 lists our references. Section 5 is the Glossary. Section 6 is the Appendices.

2 Requirements Analysis

This section provides a summary for the requirements of the Airport Process Time Simulator(APTS) application. We describe the interfaces, functional requirements, and any design constraints involved. These requirements are more detailed in the Software Requirements Specification document.

2.1 Graphical User Interface

The GUI for the APTS application shall provide an input screen where the user can choose to alter parameters of the simulation, or go with the default parameters should they not choose to change anything. It shall also provide a start button that triggers the start of the simulation.

Once the simulation has ended, there shall be a results screen where all the statistics requested by the user in the Project Requirements document shall be displayed.

2.2 Software interface

The APTS application shall utilize the Simlib library functions in order to run the simulation and calculate results. This involves invoking events based on probability, incrementing the simulation clock, storing data, and calculating the statistics requested.

2.3 Functional Requirements

This application shall simulate taxi activity at the FedEx airport primarily through the use of events advancing time within a given time period. This section summarizes the functional requirements associated with the application.

2.3.1 GUI Data Processor

The program shall take in input from the GUI, validate it, and then build a string with it to write an input file for use in the simulation. Once the simulation has finished, it shall produce a file to be read that will provide the results. This file shall be read, and parsed to relay the results to the GUI for display.

2.3.2 Computational Engine

The core of the project shall take the input file mentioned above, read it, and transform the input into a usable data structure for the simulation. Then it shall utilize Simlib to run the simulation. Once the simulations done the core shall take the results from Simlib, and calculate any results specified in the requirements document that need calculation (such as averages). Once that's done, it shall build a result file that shall be written to the user's hard drive for future viewing and also sent for display in the GUI.

2.4 Performance Requirements

The APTS application shall be efficient, easy to use, reliable, secure, and safe.

2.5 Design Constraints

This section describes customer-specified requirements that directly impact design decisions.

2.5.1 Utilize the Simlib Library

For the APTS application it has been specified in the project requirements that for the simulation we are to utilize the Simlib simulation library. The problem statement for the project at hand includes statements that are specific to Simlib, such as the use of "streams" to record data on airplanes.

2.5.2 Delete and Insert functions

The customer has specified that our project shall provide Insert and Delete functions that interact with the "Event List" in Simlib. It has also been specified that both these functions shall be written in C.

2.5.3 Provided Parameters for Simulation

The customer has specified exact parameters for the simulation with which we have implemented through the use of default parameters. There was also specification that we shall include an extra scenario where additional airplanes are included into the simulation that behave differently as well as specified extra "streams" for simlib to use on this simulation.

3 Design

3.1 Context Diagram (with description)

This section shows the context diagram for Airport Process Time Simulator (APTS). This diagram is at the highest level and shows how the various external entities (Basic Users, Advanced Users, Airport Facility, FedEx) interact with the system as a whole.

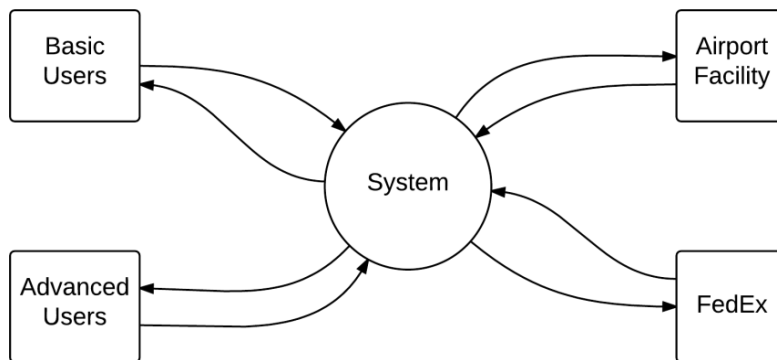


Figure 1. Context Diagram

3.2 Level 0 Data Flow Diagram and P-specs

The level 0 diagram has is the highest level of the DFD, The user inputs data that is read by the GUI Data Processor which then writes the input to the input file data store D1. The computational engine reads D1 and uses the file to compute the simulation, then writes the output to the results file, data store D2. Finally the GUI data processor reads the results from the output file data store and displays them to the user.

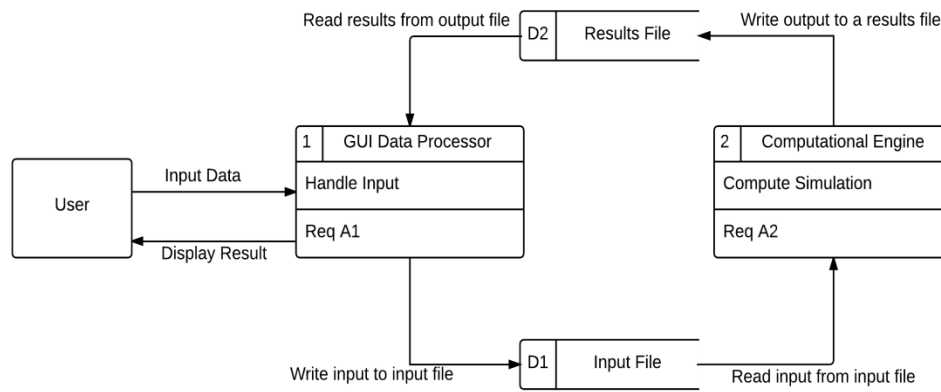


Figure 2. Level 0 DFD

1. Handle Input

/*User fills out form*/

1.1 Receive Input

1.2 Validate Input

1.3 Build Input String

1.4 Write Input File

1.5 Read Results File

1.6 Parse Results String

1.7 Display Output

2. Compute Simulation

/* User presses Start and all forms are filled out correctly*/

2.1 Read Input File

2.2 Process Input

2.3 Simlib Process

2.3.1 Initialize

2.3.2 Insert Event

2.3.3 Evaluate Event

2.3.4 Delete Event

2.3.5 Update Event

2.4 Calculate Results

2.5 Build Results String

2.6 Write Results File

3.3 Level 1/2 Data Flow Diagrams (with P-specs as necessary)

This section decomposes and details processes 1 and 2 in the level 0 DFD. Figure 3, pictured below expands process 1- Handle Input. 1) The user inputs a form which is received by the GUI data processor. 2) After the input is received it is then validated. 3) If the input is valid, it is written to the input file. If not valid the GUI displays “Invalid input”. 4) After the simulation is run, the results file is read, turned into a string and then parsed so that it can be displayed to the user.

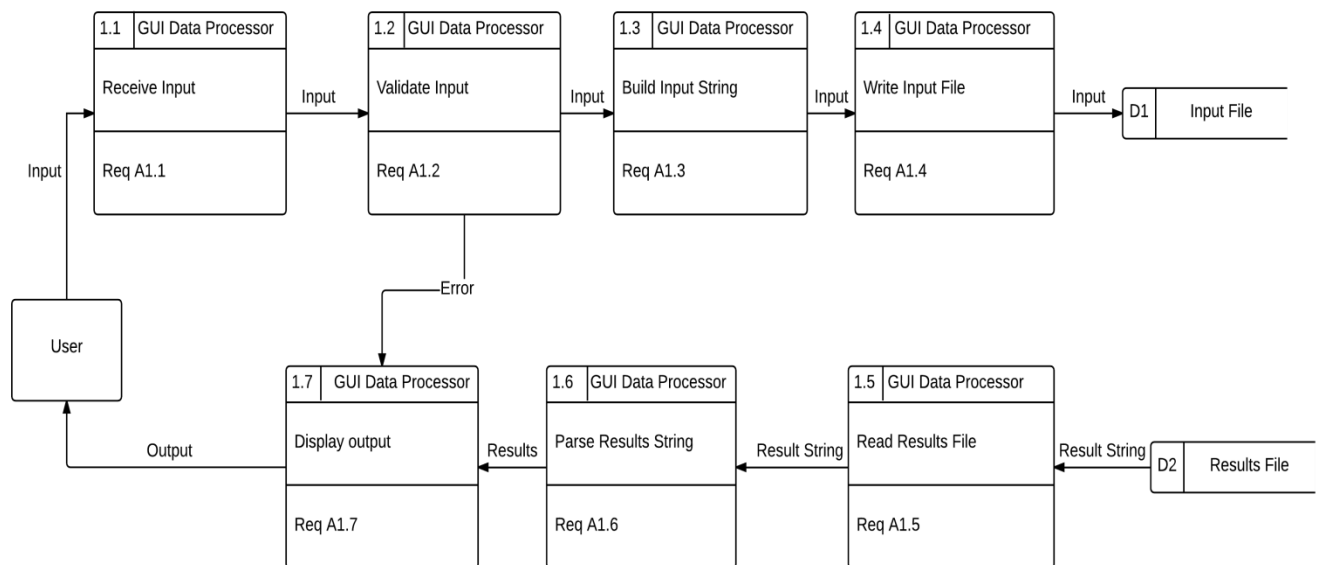


Figure 3. Level 1(1.n) DFD

Figure 4, pictured below expands process 2- Compute Simulation. 1) The input file is read by the Computational Engine, processed and then handled by Simlib. 2) The raw simulation data is then calculated and the results are turned into a results string. 4) The results string is written to the results file.

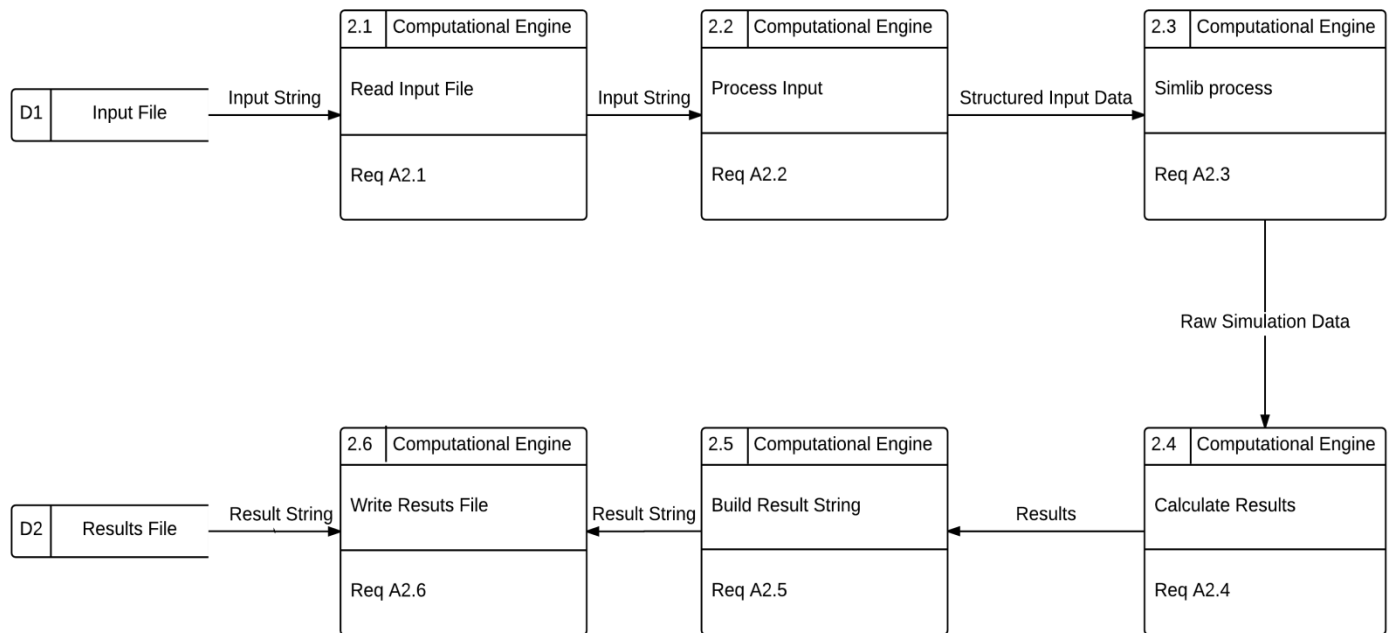


Figure 4. Level 1(2.n) DFD

Figure 5, pictured below expands process 2.3- the Simlib process. 1) Inside simlib, an event is initialized and then added to the queue. 2) Once inside the queue, the event will be updated, or deleted once completed.

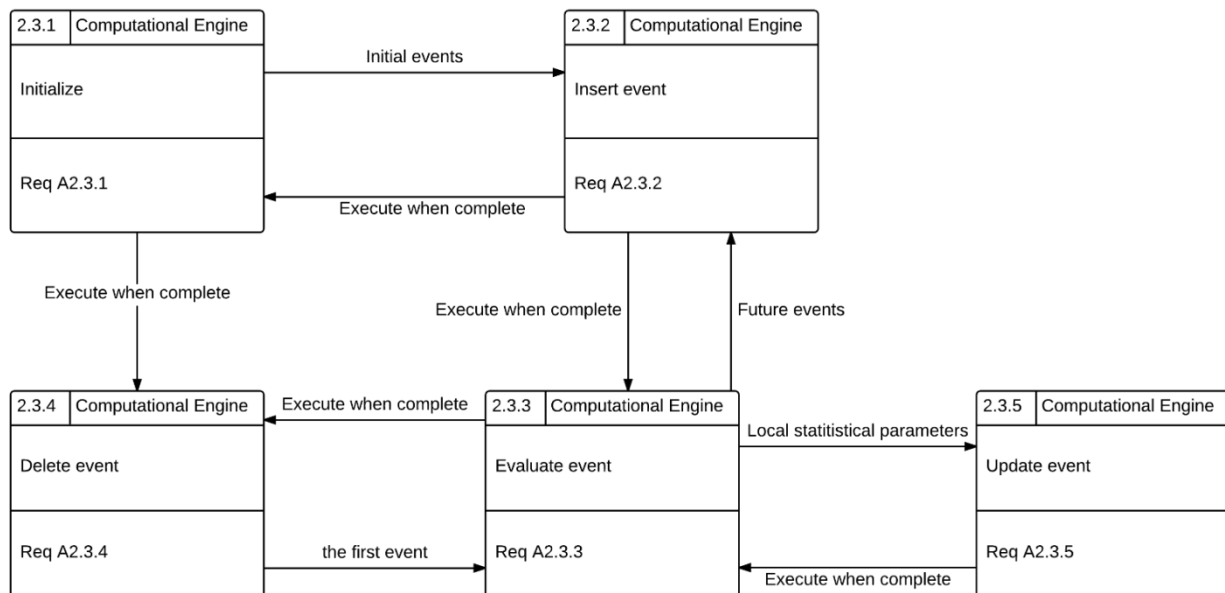


Figure 5. Level 2 (2.3.n) DFD

3.4 Transaction Analysis In this section we will take our system and split it into a system of interconnected subsystems. Our test cases to verify our functional requirements will be able to test parts of the system and with further. Our transactions, will be detailed in section 3.4.2.

3.4.1 Partitions of the System DFD

Our DFD is comprised of 3 levels and it is combined below into one DFD shown below.

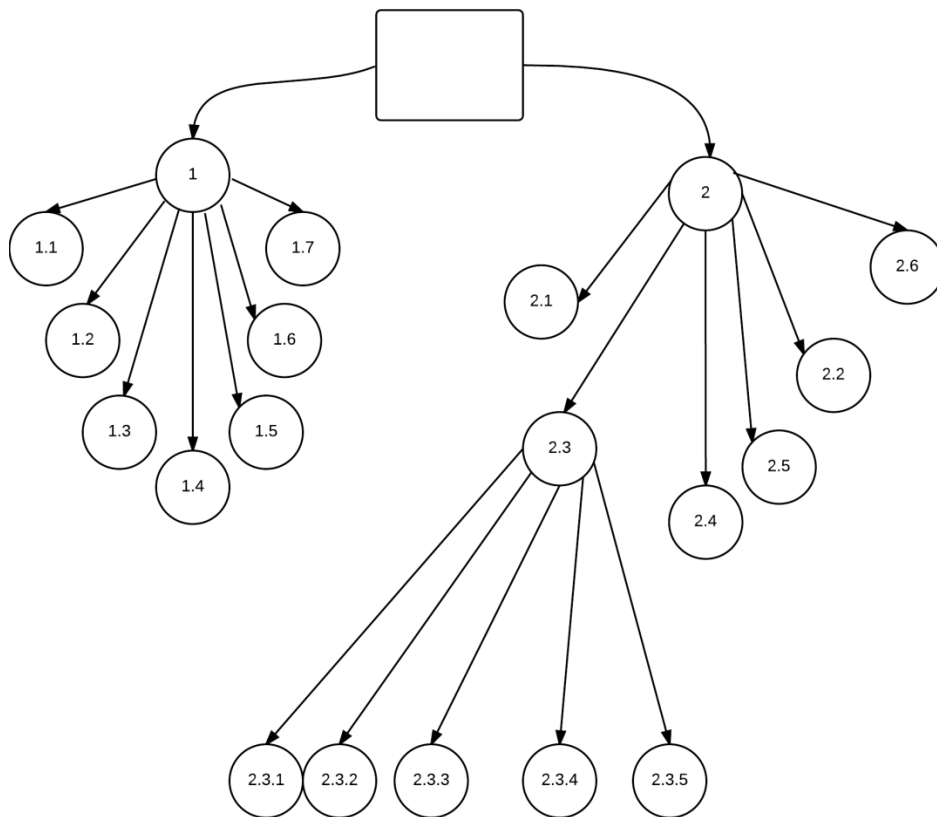


Figure 6. System DFD Partition

3.4.2 Identified Transactions

This sections lists our system transactions in the following format:

- The event in the systems environment that causes the transaction to occur;
- The stimulus that is applied to the system to inform it about the event;
- The activity that is performed by the system as a result of the stimulus;
- The response that this generates in terms of output from the system;

- The effect that this has upon the environment;

Transaction 1

Event: Run basic scenario

Stimulus: User presses start.

Activity: System runs simulation using default parameters

Response: Results are displayed

Effect: Verify successful simulation

Transaction 2

Event: Run advanced user scenario

Stimulus: User fills out fills that provide additional parameters for the simulation and presses start

Activity: System runs simulation using specified parameters

Response: Results are displayed

Effect: Verifies the following: Valid input, successful simulation with additional parameters

Transaction 3

Event: Run advanced scenario

Stimulus: User fills form incorrectly and presses start

Activity: System does not run simulation

Response: Display “Invalid Input”

Effect: Verify the checking of input parameters

3.5 Transform Analysis and Structure Chart

The transform analysis of our system shown below comes directly from the DFD's in a compact, concise hierarchal structure that shows the flow from beginning to end of our program's main process. As shown in transaction 3 in section 3.4.2 our simulation never terminates once started and will always start as long as the input parameters are correct.

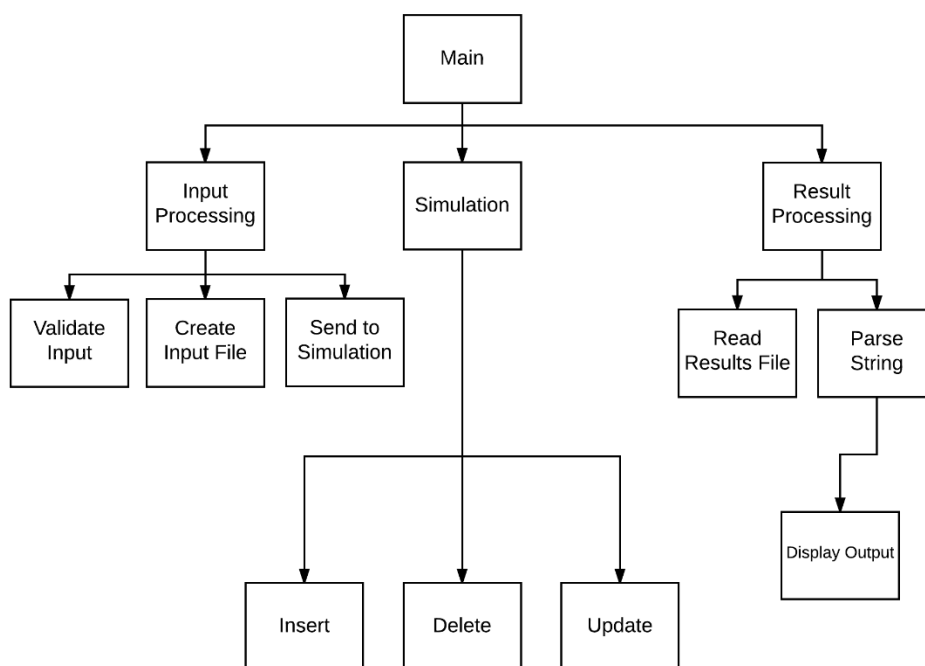


Figure 7. APTS Structure Chart

4 References

1. "Diagrams Done Right." *Flow Chart Maker & Online Diagram Software*. N.p., n.d. Web. 12 Sept. 2014.
2. Team Bitrunners, "SRS" October, 2014.

5 Glossary

This section gives definitions for certain terms used in this document.

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

Appendix A GUI Prototype






Simulation Length:		Storms:		Taxiway:	
	Length of simulation: <input type="text"/>		Mean of storm occurrence: <input type="text"/> Base Length: <input type="text"/> Variation Length: <input type="text"/>		Number of Taxiways: <input type="text"/> Travel Time: <input type="text"/>
Fixed planes:		External planes:			
	Arrival rate:	<input type="text"/>		Number of sets of external planes:	<input type="text"/>
	Base Length:	<input type="text"/>		Number of planes in the set:	<input type="text"/>
	Variation Length:	<input type="text"/>		Cat3 landing gear probability:	<input type="text"/>
	Number of planes:	<input type="text"/>		Variation in loading time:	<input type="text"/>
	Frequency:	<input type="text"/>		Frequency:	<input type="text"/>
	Base loading time:	<input type="text"/>		Base loading time:	<input type="text"/>
	Variation in loading time:	<input type="text"/>		Variation in loading time:	<input type="text"/>
	Cat 3 landing gear probability:	<input type="text"/>		Cat 3 landing gear probability:	<input type="text"/>
<input type="button" value="Submit"/>					

Figure 8. GUI Prototype

Appendix B Project Schedule

Table 1. Project Schedule

Start Date	Tasks
9/8	Complete Project Plan. Setup Linux environment. Learn git and GitHub.
9/15	Install Simlib. Start working on first three requirements. Decide on unit testing framework.
9/22	Create GUI. Discuss test cases for the project.
9/24	Project Plan due
9/29	Work on PDR. Review GUI design and make changes.
10/6	Finish up PDR. Review progress of the project and code.
10/10	PDR presentation due
10/11	Work on SRS
10/12	SRS due
10/13	Work on CDR and design notebook. Finish up code.
10/19	CDR presentation due
10/25	Finish design notebook
10/26	Design notebook due
10/27	Finish GUI code. Start testing. Modify test cases. Discuss integration testing.
11/16	Test Report

11/23 User Manual

12/9 Final Demo

Appendix C Requirements Traceability Matrix

Table 2 Requirements Traceability Matrix

Req. ID System Level.	Req. ID Sub-system level	Req. ID Sub-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	
		A2.3.1	2.3.1	Initialize	I	
		A2.3.2	2.3.2	Insert Event	T/D	
		A2.3.3	2.3.3	Evaluate Event	A,I	
		A2.3.4	2.3.4	Delete Event	T/D	

		A2.3.5	2.3.5	Update Event	T/D	
	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	

Appendix D Identified Test Cases

The test cases in this section specifically will be based on the transactions in section 3.4. They will test that each scenario works as intended.

Test Case 1:

Purpose: Verify system with default parameters.

Method: User presses start simulation as default user.

Response: Results are displayed to output GUI.

Test Case 2:

Purpose: Verify system runs with advanced parameters.

Method: User presses start simulation after altering input parameters as an advanced user.

Response: Results are displayed to output GUI.

Test Case 3:

Purpose: Verify system that system doesn't run with incorrect parameters.

Method: User presses start simulation after altering input parameters incorrectly as an advanced user.

Response: GUI displays "Invalid Input".