Project Plan

For

Global Package Courier Tracking

COMP 4081 Software Engineering

(23 September 2014)

BitRunners (Team #5)

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

Stephen Moo-Young

Version 1.3

Abstract

The project plan details our approach to the methodology used for developing a proper simulation for global FedEx shipping. We outline a plan to solve the problem of taxi time occupancy under various constraints by simulating these conditions. In here we also throw some light on the division of labor amongst our group. This document further explains how we plan to implement our project and includes a preliminary workflow. We have tried to address the issue of time management in the schedule. We made the schedule detailed enough to remain on track. A small plan including the types of test has been outlined which will develop upon in further documents. Finally we provide risk analysis and our plans to mitigate any contingency.

The project plan consists of:

- Description of the project.
- Our design process.
- Preliminary context diagram.
- Coding standards.
- Testing phase.
- Possible Risks

Table of Contents

1 Introduction 1

1.1 Problem Description 1

1.2 Goal 1

1.3 Scope 1

1.4 Definitions, acronyms and abbreviations 2

2 Process 3

2.1 Objectives and Milestones 3

2.2 Project Organization (org and roles) 3

2.3 Work Breakdown 4

2.4 Hardware and software requirements 4

2.5 Project Schedule 5

3 Project Validation and Verification 6

(Testing strategy and traceability approach)

4 Risk Analysis 6

5 References 6

APPENDIX A: Coding Standards 7

Table of Figure(s)	
Figure 1	*Context diagram*

List of Tables

Table 1: Members role and name

Table 2: Tentative Schedule

1 Introduction

Software simulation has been proved to provide insightful analysis of various physical systems. Through this project we explore and try to get involved in the fascinating world of computer modeling of the freighting industry.

1.1 Problem Description

Problems of logistics can be said to be of great importance to FedEx. The problem we are facing consists of calculating the time percentages various components of the airport are occupied with. For this project we assume events happen following different random variables. These events have effects on the overall efficiency of FedEx's shipment of packages across the world. Therefore it is beneficial to simulate these events and analyze the results obtained.

1.2 Goals

The main objective is to successfully simulate the occurrence of events and their outcomes. Our final goals are calculating what fraction of time taxis and berths are occupied or idle, the effects of airplane arrivals and storms, the expected average in-port residence time for each type of airplane. Streams shall be used to record simulation parameters. Simulations will be repeated taking into account five airplanes which deliver cargo to United Kingdom.

1.3 Scope

To develop this program we will follow spiral-model. For the sake of simplification we have decided to make a web GUI. This will make the user interface of the program easy to implement and use. Figure 1 explains how the GUI will talk to the core of the program. The popular web format XML was chosen for writing the files. Once the core has the information from the GUI it can request particular simulations from Simlib and forward the outputs to the file read by the GUI.

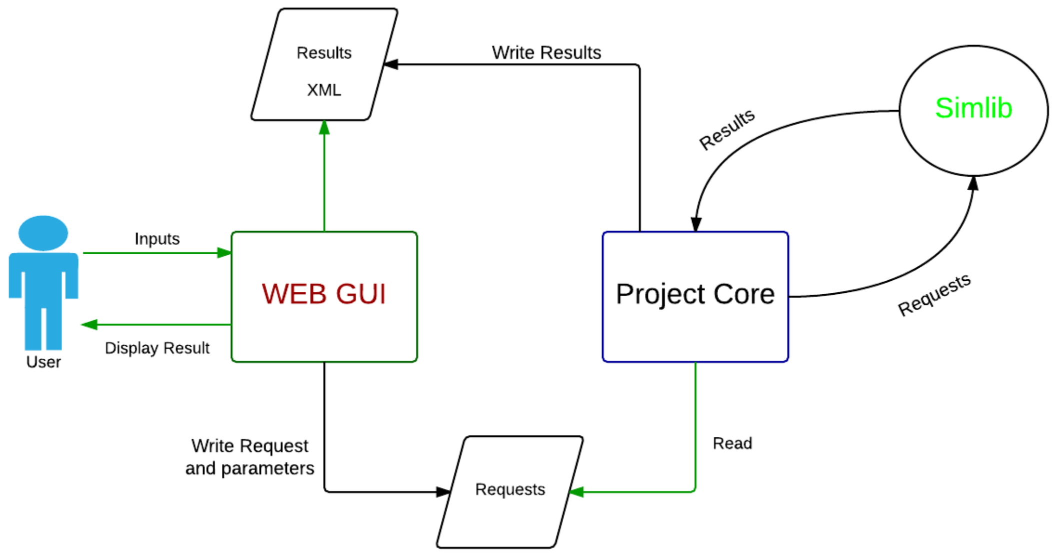

Figure 1. Context Diagram

1.4 Definitions, acronyms and abbreviations

- GUI: Graphical User Interface
- XML: eXtensible Markup Language
- API: Application Programming Interface
- Simlib: Library that provides a tool set to run simulations
- PHP: PHP Hypertext Preprocessor
- SRS: System Requirements and Specifications
- PDR: Preliminary Design Review
- CDR: Critical Design Review

2 Process

In this section we address the way we are going to carry out the project. We will go over our objectives, milestones and roles of each member.

2.1 Objectives:

We think that the following objectives are necessary for the successful implementation of this project:

- Decide which programming language is most suitable and convenient for the two parts of the project.
- Get familiar with coding standards that we adopted (Appendix A).
- Open a GitHub repository and setup environment for each member of the team.
- Gain preliminary knowledge of Simlib by completing the second and the third requirements,
- Hold at least one meeting per week to cover the progress of the project.
- Do everything possible to meet deadlines and complete the project within the assigned time span.

2.2 Milestones

In order to carry out our objectives we have set ourselves the following milestones whose detailed timing will be provided in the schedule:

- Complete the project plan
- Understand Simlib and its API
- Create a suitable GUI
- Create a bridge between Simlib and the GUI
- Write a project report that explains our design.

2.3 Project Organization

The following are the roles assigned to each team member:

Table 1: Members role and name

Name	Role
Dereje Arega	Software Engineer
Ismael Alonso	Proj. Lead
Stephen Moo-Young	Lead Test Engr
Chris Hubbard	Requirements Engr
Ashlesh Gawande	Lead Programmer
Matthew Longley	TBD

2.4 Work Breakdown

Each lead will designate particular tasks dynamically at their convenience.

- Dereje Arega: Documents meetings and assists programmers.
- Ismael Alonso: Schedules meetings and team leader.
- Stephen Moo-Young: Creates test cases for code and responsible for proper syntax.
- Chris Hubbard: Makes sure requirements are met and assist other members.
- Ashlesh Gawande: Designates code assignments and responsible for deadlines.
- Matthew Longley: TBD

2.5 Hardware and software requirements

The following requirements are necessary to successfully complete our objectives.

Software

- Eclipse for C/C++
- Simlib
- Web GUI (Apache server, PHP)
- Private repository on GitHub, git as a content revision system.

Hardware

- Server capable computer

2.5 Project Schedule

We will keep our project on track by using Table 2:

Table 2: Tentative Schedule

Start Date	End Date	Tasks
9/8	9/12	Complete Project Plan. Setup Linux environment. Learn git and GitHub.
9/15	9/19	Install Symlib. Start working on first three requirements. Decide on unit testing framework.
9/22	9/26	Create GUI. Finish SRS. Discuss test cases for the project.
9/29	10/3	Work on PDR. Review GUI design and make changes.
10/6	10/10	Finish up PDR. Review progress of the project and code.
10/13	10/17	Work on CDR and design notebook. Finish up code.
10/20	10/24	Finish up design notebook. Practice presentation.
10/27	11/16	Finish GUI code. Start documentation. Start testing. Modify test cases. Discuss integration testing.
11/19	11/23	Complete user manual. Check coding convention is right (Appendix A).
11/23	12/9	Run simulation and finish project. Run test cases.

3 Project Validation and Testing

Throughout our project periodic testing is necessary to insure correct functionality of the program. Here are some of the tests we plan to implement:

- Interface Testing: The GUI will be tested for reliability and ease of use.
- Unit testing for Project Core
- Integration testing for every major update to the code
- Deriving a methodology to validate the output of the program.

4 Risk Analysis

The following are possible issues that we may encounter with our project

- Not finding Proper Documentation of Simlib
    - Going through the books provided and making notes
- Familiarity of C++
    - Our project leader who has familiarity with C++ will provide occasional tutoring to the members who need it
- Proper group communication
    - Meetings will be announced through email and text
    - Each member will present his progress on the assigned task

5 References

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

All future tools will be cited dynamically as we use them.

Appendix A. Coding Standards

C++ code convention with camel case for naming