ProCP Process Report

11 Jun 2015

Prepared by: Group B - Stoyan Staynov  
 Dan Velev  
 Boris Mihailovic  
 Sophia Njenga  
 Veselin Kichukov  
 Nikolay Polyanov

Traffic Lights Simulation

# EXECUTIVE SUMMARY

## The System

The city of Sharp requires a traffic light simulation program. They wish to use the program to simulate traffic around the city so as to increase the efficiency of the transport network. The system allows you to simulate the traffic of various crossings, by increasing the number of cars at crossings and changing the frequency of the changing traffic lights.

## The Class Diagram

The class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Before implementation of code, it is very important to make a class diagram, it helps you understand the application as a whole, the methods and attributes, the relationships between classes.

## The Functional Requirement

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

Traffic lights Simulation

A commonplace occurrence in software development is to not give the customer the completed application with all functionality present in one go, this would be, from a business standpoint, not that beneficial to the team in the long-term. In order to prioritize the functionalities, we use the functional requirements to build a ‘MOSCOW’ system (Must, Should, Could, Would). For the first version of the application, we would implement the Must’s, later versions, Should etc.

## The Use Case

A use case is a list of steps, typically defining interactions between a role (an "actor") and a system, to achieve a goal.

This first step is very important because it helped us visualize our application, the functionalities, how these functionalities could be implemented at a very basic level. The use cases also gave us the basic building blocks to form our MOSCOW. It is important to note that by the implementation stage the use cases had been slightly modified.

# Project management

## Tool

We did not use a project management tool.\* We did however use Dropbox to share our work, in this way everybody had the current working copy of the project plan, use cases, class diagrams etc. We also used Facebook chat as our main method of communication outside of school hours.

## Meeting

Scheduling a meeting within our group was very tricky as we all have different schedules and commitments to other projects however we found a session in which we could all be free. We had group meetings every week during the 12.25 - 13.15 lunch session to discuss the project and every Wednesday at least 3 hours to work on the project. During the meetings we would divide the work and work out some deadlines for submission. Some assignments required us to work in small groups, we used this time to complete these assignments

Traffic lights simulation

## Task Board

Below is the task board which provides the scheduling of tasks and those responsible for each task.

| SPRINT | NAME AND DESCRIPTION | TASKS | PERSON RESPONSIBLE |
| --- | --- | --- | --- |
| **1** | Project Plan | Project Statement | Boris / Nickolay |
|  |  | Project Phasing | Stoyan / Sophia |
|  |  | Project Risks | Sophia / Veselin |
|  |  | Project Constraints | Sophia / Stoyan / Dan |
|  |  | MOSQUITO | Dan / Sophia |
|  |  | Organization | Boris / Veselin |
| **2** | Create URS document | Use Cases | All group members |
|  |  | Requirements and rules | Boris / Veselin |
|  |  | MOSCOW | Nickolay / Boris |
|  | Create Test plan and Test Cases |  | Dan / Veselin / Stoyan |
| **3** | Create the design document | Class Diagram | All group members |
|  |  | Sequence diagrams | Dan / Stoyan |
|  |  | Modify class diagram | Dan / Nickolay |
| **4** | Implementing the code | Cars / Car Movement / Group / Form1 / Lane / Direction / Simulation / Traffic Light | Dan |
|  |  | Traffic Light / Crossing / Form1 / Light Color / Form2 | Stoyan |
|  |  | Car / Car Movement | Sophia |
|  |  | Group / Simulation | Boris |
|  |  | Lane / FileHelper | Veselin |
|  | Create presentation |  | Boris / Nickolay |
|  | Create process report |  | Sophia |
|  | Test code |  | Dan / Veselin / Stoyan |

# Decision Making

## GitHub and Dropbox

GitHub was our primary choice for version control during implementation, however during the realization the project we used Dropbox as our primary version control and sharing service. GitHub is the most popular web-based Git repository hosting service while Dropbox is the most popular file synchronization, cloud storage and file sharing tool. We chose Dropbox because it was already widely used among the group members, it is simple and quick to use and it has a simple version control system. GitHub was our first choice for implementation of code because of its simple to use applications and setting up.

## Choosing the GUI and controls

Having discussed the various ways in which we could implement the application, we decided on a simple design which could be used by anyone without prior knowledge of the applications controls. Our first GUI draft was very basic but gave us an idea of what and how we could implement most functions.

## Creating the use cases and class diagram

Since use cases are what make up most of the MOSCOW ‘must’ requirements, we were very careful in choosing them. Each group member was assigned two use cases and with the help of the GUI draft design, draw out clear steps towards developing a function. The class diagram required the whole group’s attention, there were a lot of modifications and discussions on the classes required and methods within each class.

# Problems & Solutions

## Version control

This was the first time for any of us to use GitHub for versioning, as a result we had a lot of problems at the beginning. The most frequent problem was conflicts arising from committing and syncing the same file by multiple people. We went through the GitHub tutorials, help documentation online and of course the stackoverflow.com website for solutions. Thankfully, we found solutions to most of the problems we were facing with versioning.

## Designing

Although we had the use cases, class diagram and sequence diagrams to help us with implementing the application we still had issues visualizing the whole application. During implementation, we changed the class diagram to suit our code’s functionality.

## Teamwork

Our group experienced some of the problems associated with teamwork; lack of motivation, difficulty making decisions, scheduling issues. However, we overcame these problems with constant communication through informal chats on Facebook and Skype. These chats helped a lot as we were able to discuss the project without having to meet in person. To increase motivation within the group we would place deadlines on individual assignments.