3/1/2016

# User Requirements specifications

# Traffic Lights

# Final Version

Monica Stoica Rosen Danev Alexandru Vinerean Blagovest Tsarev Dymtro Bunin Preslav Ghercev Dimitar Vikentiev Ventsislav Yotov

fONTYS UNIVERSITY OF APPLIED SCIENCE

Contents

[User Requirements specifications 0](file://localhost/Users/user/Desktop/ProCP/In%20progress/User%20requirements%20specificationsVersion%20I.docx#_Toc445200834)

[Traffic Lights 0](file://localhost/Users/user/Desktop/ProCP/In%20progress/User%20requirements%20specificationsVersion%20I.docx#_Toc445200835)

[Version II 0](file://localhost/Users/user/Desktop/ProCP/In%20progress/User%20requirements%20specificationsVersion%20I.docx#_Toc445200836)

[Introduction 2](#_Toc445200837)

[Functional requirements 3](#_Toc445200838)

[(use-cases) 3](#_Toc445200839)

[MoSCoW 9](#_Toc445200840)

[Non-functional requirements 10](#_Toc445200841)

[Appendix A 11](#_Toc445200842)

[Traffic light components 11](#_Toc445200843)

# 

# Introduction

Our group consists of six members: Rosen Danev, Blagovest Tsarev,Monica Stoica, Alexandru Vinerean, Ventsislav Yotov and Dmyro Bunin students of class EI7S2.

The following document describes the implementation of an object-oriented software product using UML techniques.

The goal of this software system is to build a traffic lights simulator consisting of traffic lights, sensors and crossings. In addition, the User Requirements Specification (URS) will be described such as functional and non-functional requirements and user interface.

The functional requirements are represented by use cases and MoScoW. We have chosen the most suitable use cases so that the most functionality of our system will be covered. In this way, we were able to determine the most appropriate user-friendly interface.

Appendix A contains information about the traffic light components and how they work.

# Functional requirements

# (use-cases)

*The following use-cases represent the functional requirements that the traffic light simulator software will be providing.*

**Goal:** Add a crossing.

**Pre:** There is no other crossing drown

**Actor**: User of the system

**MSS**:

1. The actor selects the one of the two crossing types by clicking on it
2. The actor clicks on the screen where he/she wants to place it
3. The system adds the element to the internal structure and draws it on the screen

**Extensions:**

**2a**. There is already a crossing on the same spot. The system displays an error and the use case ends.

**3.a** The newly created crossing is connected to an already existing crossing.

**1.** The system updates the internal structure by setting the flow of the new crossing equal to the flow of the already exiting crossing (depending on the side that connects them).

**Goal:** Remove a crossing

**Pre:** There is at least one crossing on the drawing board

**Actor**: User of the system

**MSS**:

1. The actor selects the crossing he/she wants to remove by double clicking on it

2. The user presses the ‘Delete’ button

3. The system removes the element from the internal structure and removes the selected crossing

**Goal:** Add an element (sensor or traffic light)

**Pre:** There is at least one crossing drawn

**Actor**: User of the system

**MSS**:

1. The actor selects the element he/she wants to add by clicking on it
2. The actor clicks on a spot on the screen where he/she wants to place the element
3. The system adds the element to the internal structure and draws it on the screen

**Extension:**

**2a.** There is already an element on that spot. The system displays an error and the use case ends

**3a.**  If the element is a traffic light/sensor for pedestrians and there is no crossing drawn, the system displays an error and the use case ends.

**Goal:** Remove an element.

**Pre:** There is at least one element on the drawing board

**Actor**: User of the system

**MSS**:

1. The actor selects the element he/she wants to delete by double clicking on it.
2. The actor presses the ‘Delete’ button
3. The system removes the element from the internal structure and from the screen.

**Extension:**

**1a.** The element is a pedestrian crossing. The system will delete all elements related to that crossing (pedestrian lights/sensors)

**Goal:** Save a file

**Actors**: User of the system

**MSS:**

1. The actor presses the ‘Save’ button.
2. The system displays the time and date of the last save in a label informing the actor that the save is done

**Extensions:**

1a.The file has no location or name on the disk then the system displays a dialog box asking the actor if he wants to save it.

1. b If the actor wants to save it, he is sent to ‘Save as file’ use case, step 2. If not, the use case ends.

**Goal:** ‘Save as’ file

**Actors**: User of the system

**MSS:**

1. The actor presses the ‘Save as’ button.
2. The system displays a dialog box.
3. The actor chooses a location.
4. The actor chooses a name for the file.
5. The actor confirms by clicking the Save button.
6. The system saves the changes
7. The system closes the dialog box.

**Extensions:**

**2a.**The actor presses the ‘Cancel’ button and the use case ends.

**4a.**There is already a traffic light file with that name

1. The system displays a warning and asking the actor if he/she wants to override the exiting file. If yes, the use case continues. If not, the actor is returned to step 4.

**Goal:** Open an existing file

**Actors**: User of the system

**MSS:**

1. The actor presses the ‘Load file’ button.
2. The system displays a dialog box.
3. The actor presses the browse button and selects the the traffic light file he/she wishes to open
4. The actor confirms by clicking the Open button.
5. The system closes the dialog box.
6. The system loads the file.
7. The system displays all the information from the file.

**Extensions:**

**1a**.The system displays a dialog box asking the actor if he/she wishes to save the current file. If yes, go to use case save file and come back to step 2.

**3a**.The actor presses the ‘Cancel’ button and exits the use case

**5a**.The file is not in the correct form

1. The system displays a warning.
2. The actor is returned at MSS-step 3.

**Goal:** Change the greentime of the traffic light

**Actor:** User of the system

**MSS:**

1. The actor chooses a traffic light by double clicking on it
2. The actor presses the ‘change green time’ button.
3. The system displays a dialog box.
4. The actor inputs new time and confirms the change.
5. The system updates the internal structure and changes the green time.

**Extensions:**

**4a**.The input of the actor is not an integer.

1. The system displays a warning
2. The actor is return to MSS-step 3

**4b.** The input of the actor is out of range

1. The system displays a warning
2. The actor is return to MSS-step 3

**4c**.The actor presses the ‘Cancel’ button and exits the use case

**Goal:** Set the flow for cars

**Actors**: User of the system

**MSS:**

1. The actor chooses a crossing.
2. The actor choses a lane
3. The actor presses the ‘change flow’ button.
4. The system displays a dialog box.
5. The actor chooses cars
6. The actor inputs the new flow.
7. The actor confirms the change.
8. The system closes the dialog box.
9. The system changes the flow

**Extensions:**

**5a**.The input of actor is not an integer.

1. The system displays a warning
2. The actor is return to MSS-step 2

**5b.** The input of actor is out of range

1. The system displays a warning
2. The actor is return to MSS-step 2

**5c**.The actor presses the ‘Cancel’ button and exits the use case

**Goal:** Add pedestrians

**Pre:** The crossing contains ‘zebras’

**Actors**: User of the system

**MSS:**

1. The actor chooses a ‘zebra’
2. The actor presses the ‘Add pedestrians’ button
3. The system adds pedestrians by drawing them on the crossing

**Extensions:**

1a. The actor did not click on a zebra. The system displays an error and the use case ends

**Goal:** Start the simulation.

**Pre:** There is at least one crossing on the drawing board.

**Actor**: User of the system

**MSS**:

1. The actor presses the ‘Start’ button.
2. The system starts the timer for the lights and sensors
3. The system displays the flow of cars/pedestrians is displayed on the screen

**Extension:**

1a. If the simulation was paused before this action, the animation will continue where it was paused

**Goal:** Pause the simulation.

**Actor**: User of the system

**MSS**:

1. The actor presses the ‘Pause’ button

2. The system will pause the traffic flow simulation

**Extension:**

1a. If there is no crossroad on the board the system will display a warning message

1b. If the simulation was not started the system will display a warning message and the use case ends

**Goal:** Clear the screen

**Actors**: User of the system

**MSS:**

1. The actor presses the ‘Clear’ button.
2. The system displays the dialog box asking the user to confirm or cancel
3. The system deletes everything from the drawing board

**Extensions:**

**1a**.If there is something on the drawing board and not saved, the system asks if the user wants to save it. If yes, go to use case ‘Save as’

**2**. The actor presses the ‘Cancel’ button and exits the use case

# MoSCoW

#### Must

1. **Project plan** - used to guide project team and stake holders threw out the project.
2. **User requirement specification** - describes the business needs that users require from the system.
3. **Test Plan**- describes how the product should be tested.
4. **Design Document** - describes the structure of classes by representing them as a diagram, that shows all the variables, methods and connections between classes using UML. This document also contains the GUI design
5. **GUI design** - is a picture, that represents how the application interface will look.
6. **Prototype** - incomplete version of the application, that simulates few aspect, that the final product will have.
7. **User’s manual** - describes how the application should be used by the user.
8. **Process report -** highlights how the team’s process influenced the work on the project.
9. **C# Application -** final product of our project. The application has to satisfy all of the requirements.

**The application will also fulfil the following requirements:**

* Two types of crossings possible
* Place a minimum of 1 and a maximum of 12 crossings in a grid.
* Adjust the car-streams coming from outside.
* Real-time traffic movement
* Adjust the ´green´ and the ‘red ’ time of the traffic-light
* Save changes
* Load file
* Rotate the crossing

#### Won’t

1. **Anything hardware related -** we are only delivering the application, we are not going to deliver any electronic devices for system to run on.

# Non-functional requirements

To begin with, the software system will have a user-friendly interface so that all users, no matter how experienced they are, will be able to create a simple traffic flow simulation. Also, reliability by assuring the user that the application will not crash at unexpected times. To achieve reliability, our team will deliver test plans.

The final product will be an installation file and will have a user manual with it. The source code will be delivered. We are not providing service or updates. The program won’t have any auto save feature and won’t be able to repair corrupt save files.

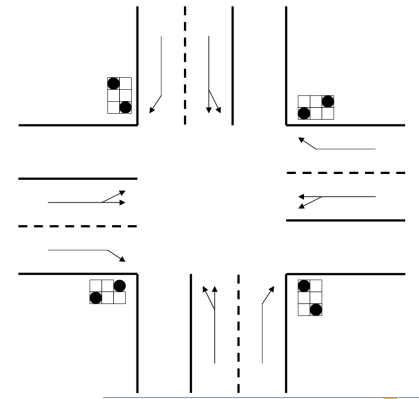
In order to run the software application, the following items are required:

* Microsoft Windows XP (Service pack 2) and further, Microsoft Vista, Microsoft Windows 7, Microsoft Windows 8, Microsoft Windows 10
* Desktop computer, notebook
* Monitor with a minimum screen resolution of 1024 x 768
* A mouse
* A keyboard

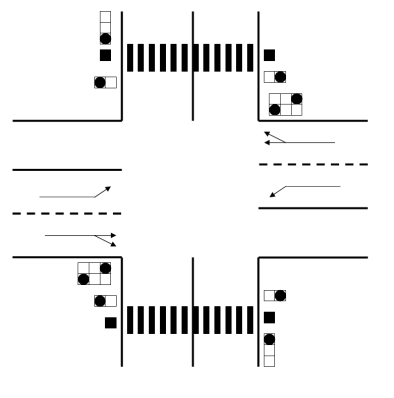
# Appendix A

# Traffic light components

The software application described above will allow the uer to create a real life traffic situation. The elements that can be added are traffic lights, sensors and crossings. Each of them has different purpose and proprieties as described below.

There are two types of junctions:

Type number one does not have any pedestrian crossings therefore the only elements that are necessary to use will be traffic lights for the cars. Each traffic light will have the function to be programmed manually-shifting the whole cycle of lights changing and giving a manually set reciprocal amount of time for the green and red light, the gap between them will be set by default to 3 seconds (yellow light)



The second type of junction has two pedestrian crossings. As we can see, that will not give a realistic model to the city’s traffic system. In order to implement a more realistic system the user will be able to rotate the junction in 180 degrees. When a pedestrian crossing is present automatically the system will place traffic lights for pedestrians as well as sensors.

Our goal is to make a randomly generated traffic and pedestrian flows. Based on a time gap of 10 seconds for instance, the system will generate a random number of vehicles for every lane in a certain range ( from 5 cars till 7 for instance) based on the time of the day(in the rush hour bigger numbers will be generated) with random speed between a certain range. The same goes for the pedestrians.

After 10 o’clock the traffic system will give priority to the first car on a certain traffic light similar to the actual traffic rules in the Netherlands.

Of course, the user will also be able to manually set the traffic flow and the time.