



**University of Applied Sciences**

# Traffic Control System

## Software Design Document

**Agnes Wadee  
Coen Stange  
Wen Li  
Yongshi Liang**

## Background and Context

This software design document provides details and decisions made for building the "Traffic control system" application. For this a graphical representation UML is used including class diagrams and sequence diagrams.

## Definitions and abbreviations

UML	Unified Modeling Language
MVVM	Model view viewmodel

## Contents

<b>1</b>	<b>System overview</b>	<b>3</b>
1.1	Abstraction layers . . . . .	3
1.1.1	Common layer . . . . .	3
1.2	View layer . . . . .	4
1.3	Business Layer . . . . .	4
1.4	Data Layer . . . . .	4
1.5	Other layers . . . . .	4
<b>2</b>	<b>Class diagrams</b>	<b>5</b>
2.0.1	Common layer . . . . .	5
2.1	View layer . . . . .	5
2.2	Business Layer . . . . .	5
2.3	Data Layer . . . . .	5
<b>3</b>	<b>Sequence diagrams</b>	<b>11</b>

# 1 System overview

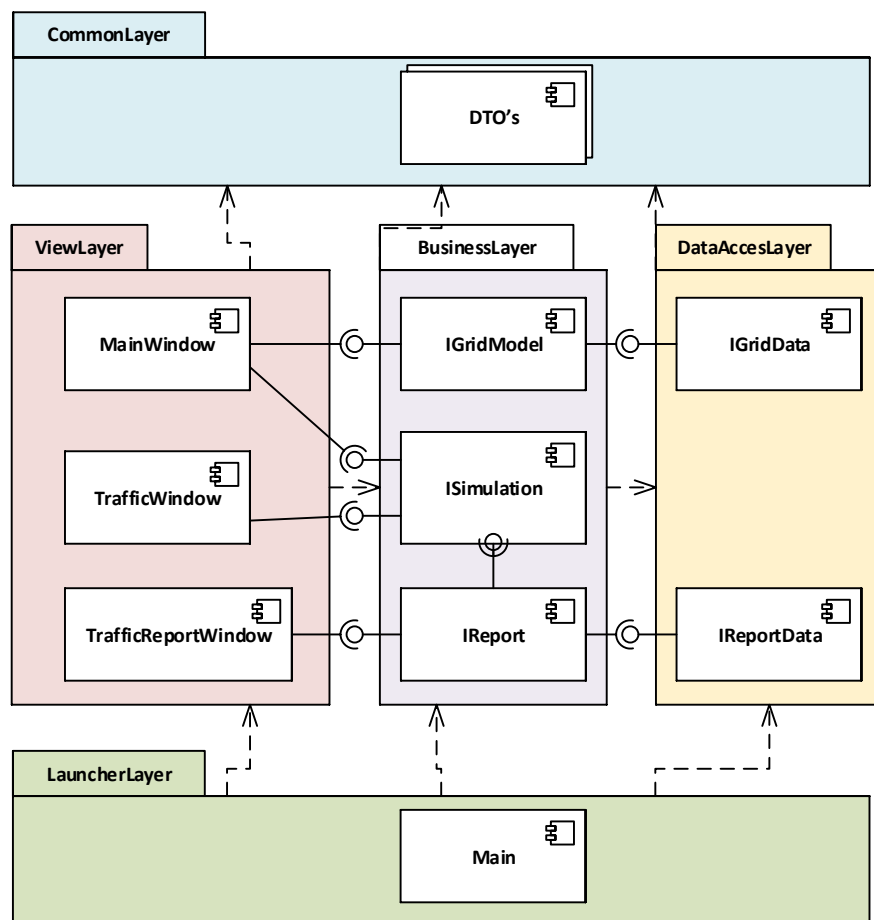


Figure 1: System overview

## 1.1 Abstraction layers

We design the software according to SOLID principles, which are common rules used for the design of the software. The classes should have only one responsibility, hence our representation logic and business logic are separated. We also use an interface driven design combined with the dependency inversion principle, hence we handle dependencies using dependency injection. By following these rules the system is more testable and abstracted.

### 1.1.1 Common layer

The **CommonLayer** can be accessed by any layer, and the layer only contains DTO's. Important about the classes of the **CommonLayer** is that they contain no logic at all, hence they only have data. The purpose for this is for easy serialization and a easy way to pass the data through all the layers.

## 1.2 View layer

Inside of the `ViewLayer` is all the representation logic, this layer handles showing an user interface. For the representation the MVVM model is used, this pattern keeps the design and the code of the view separate.

## 1.3 Business Layer

This layer contains the business logic of the system, it makes sure that all the business rules are followed. Hence this is the brain of the application which is allowed to make changes to the DTO's from the `CommonLayer`.

## 1.4 Data Layer

The data layer contains the logic for serializing the objects from the `Common Layer` to another storage. For this application the objects will be serialized to a file.

## 1.5 Other layers

The application has tests and it also require a launcher layer. Because the application can not be launched from the `ViewLayer` since it has no reference to the `DataAccessLayer`. So in the launcher layer all the dependencies will be handled.

## 2 Class diagrams

### 2.1 Common layer

The `CommonLayer` is shown in figure 2. The design of the classes are in mind that it can be shared in between layers and it can be serialized (written to file). Hence the class diagram follow the constraints that it can't use any interfaces or that there is a circular reference (it has to look like a tree). Notice that the `PedestrianEntity` and the `CarEntity` are in an `ObservableCollection` and have observable attributes. When the simulation is running the view will be notified when attributes are changed.

### 2.2 View layer

In figure 5 the class diagram of the view layer is shown. Inside of the view we do not use the usual `Graphics` used in Forms application to draw on the screen but instead a `Canvas` to draw on the screen. A lot of the drawing is already handled and we have especially bind the right values.

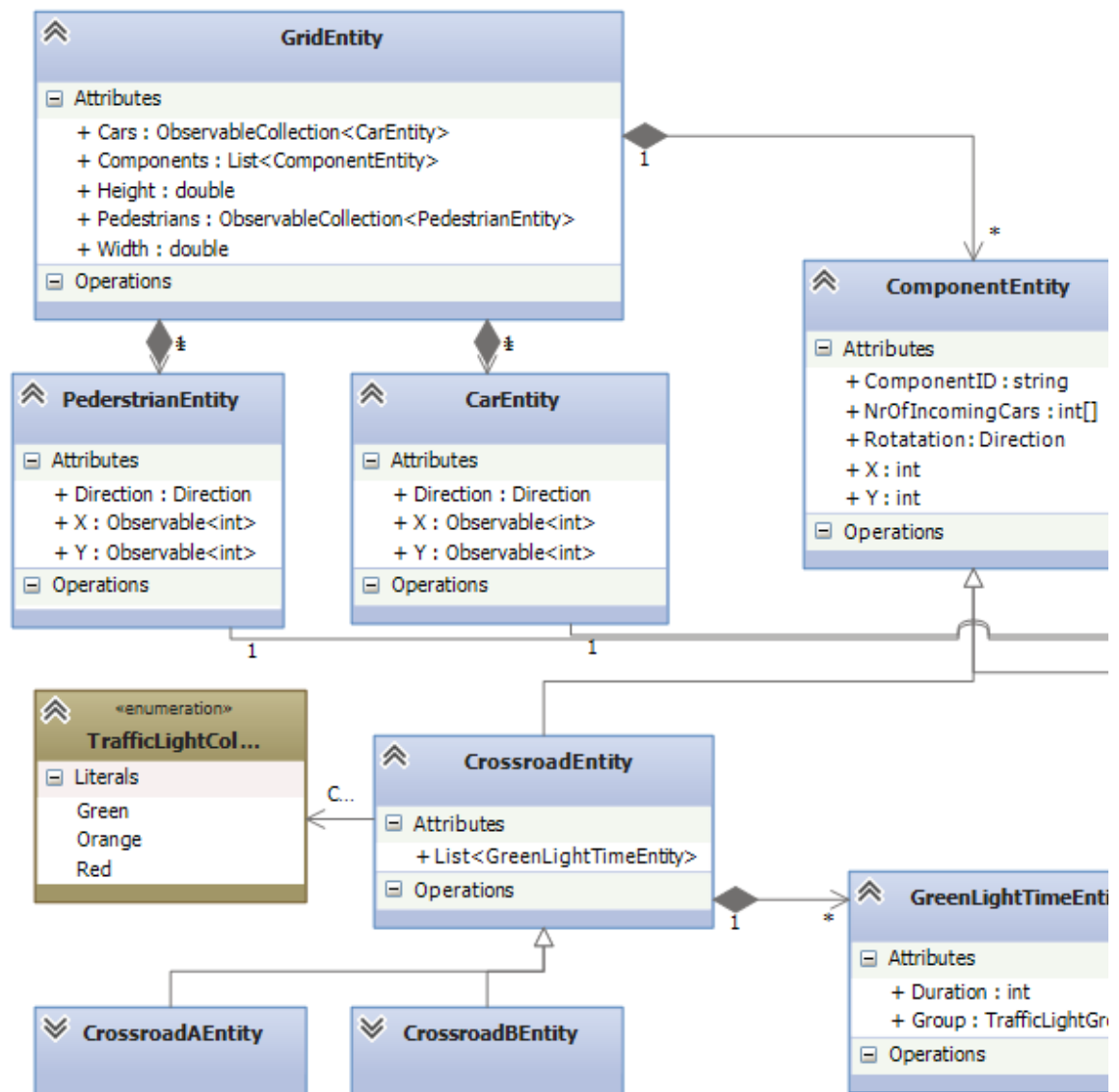
### 2.3 Business Layer

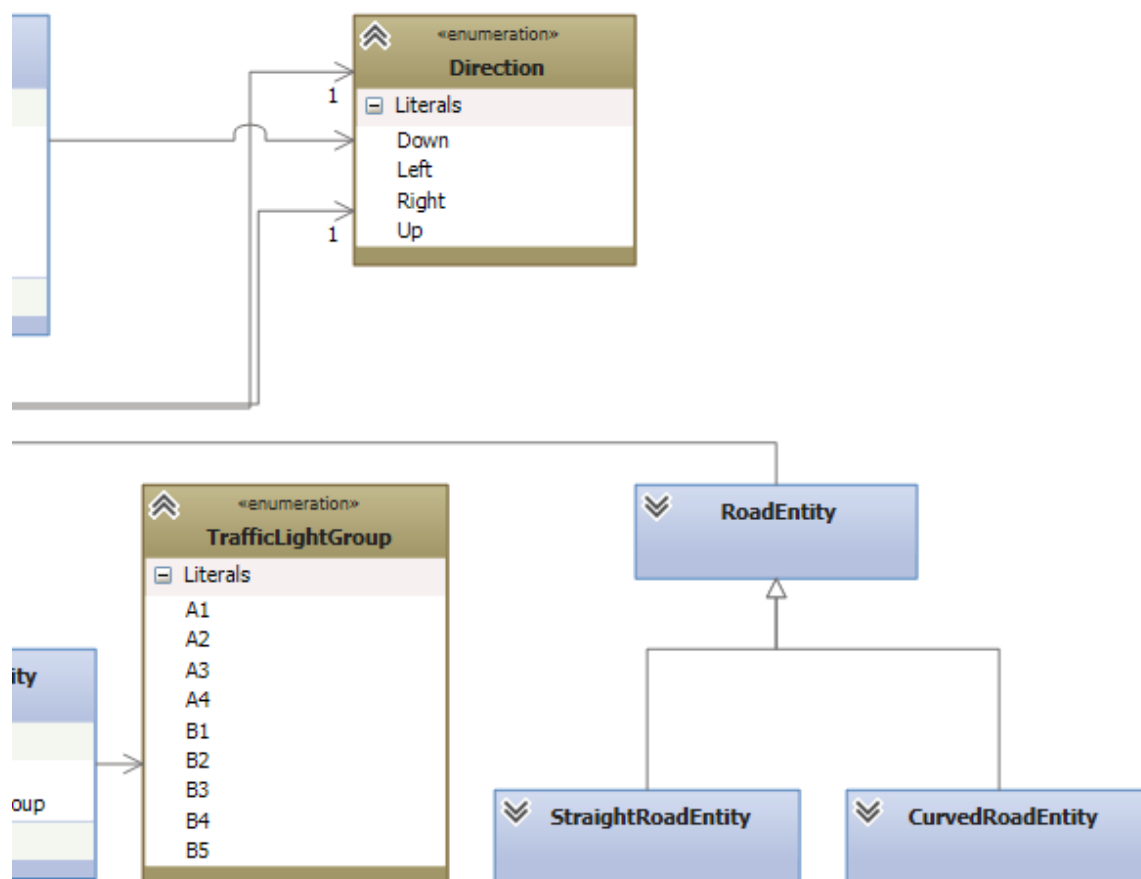
In figure 4 the class diagram of the business layer is shown. We choose to make operations for adding and deleting components instead of using an observable collection since by using operations is more simple to maintain business rules.

### 2.4 Data Layer

In figure 3 the class diagram of the data layer is shown. The data layer does not show concrete implementation since the classes don't require to have a state.

Figure 2: Class diagram common layer





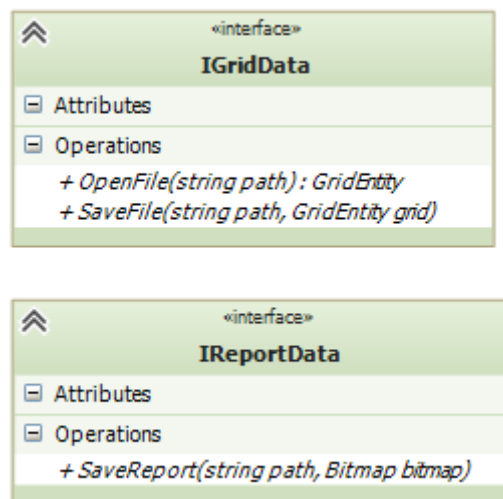


Figure 3: Class diagram data access layer



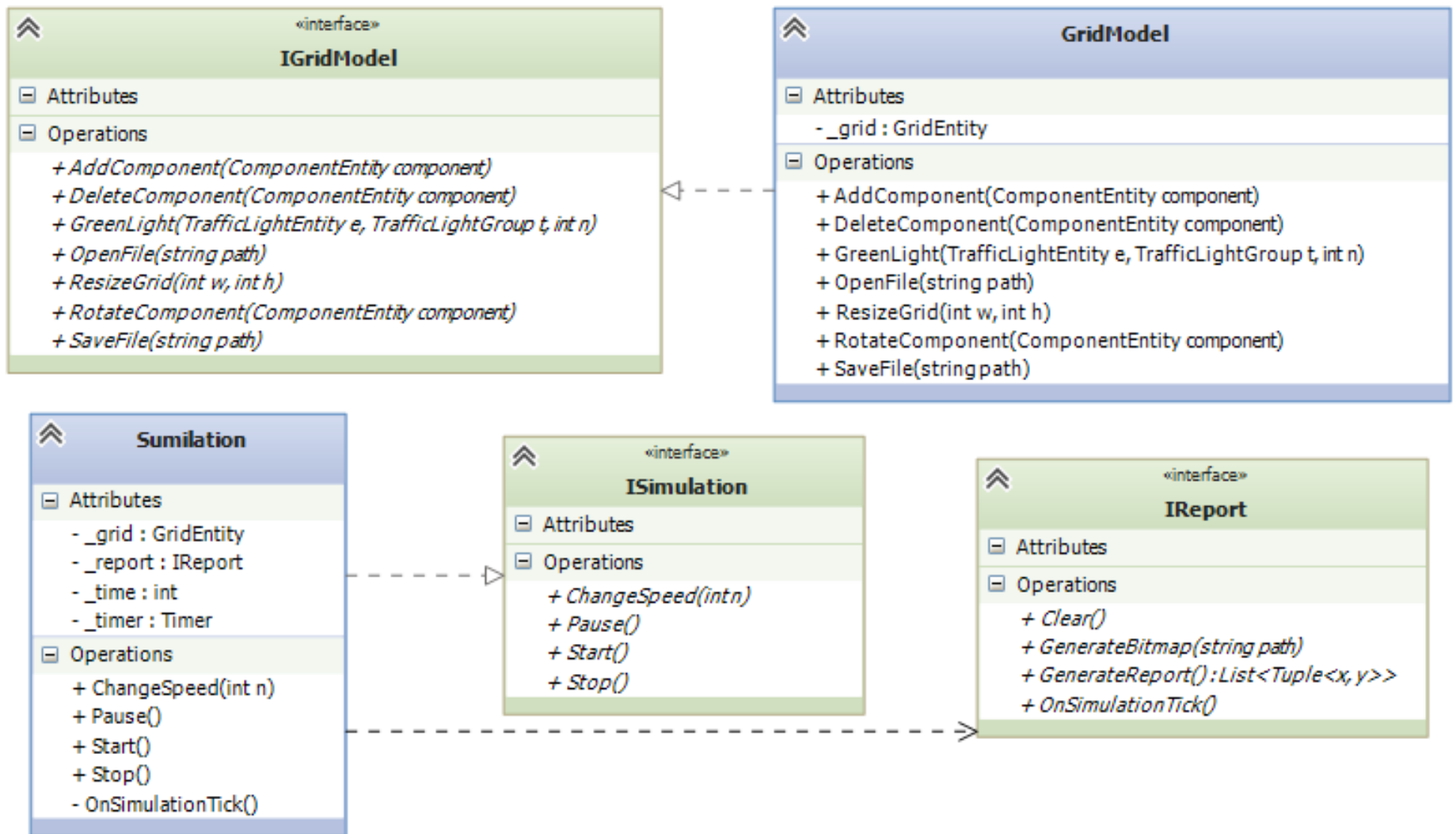


Figure 4: Class diagram business layer

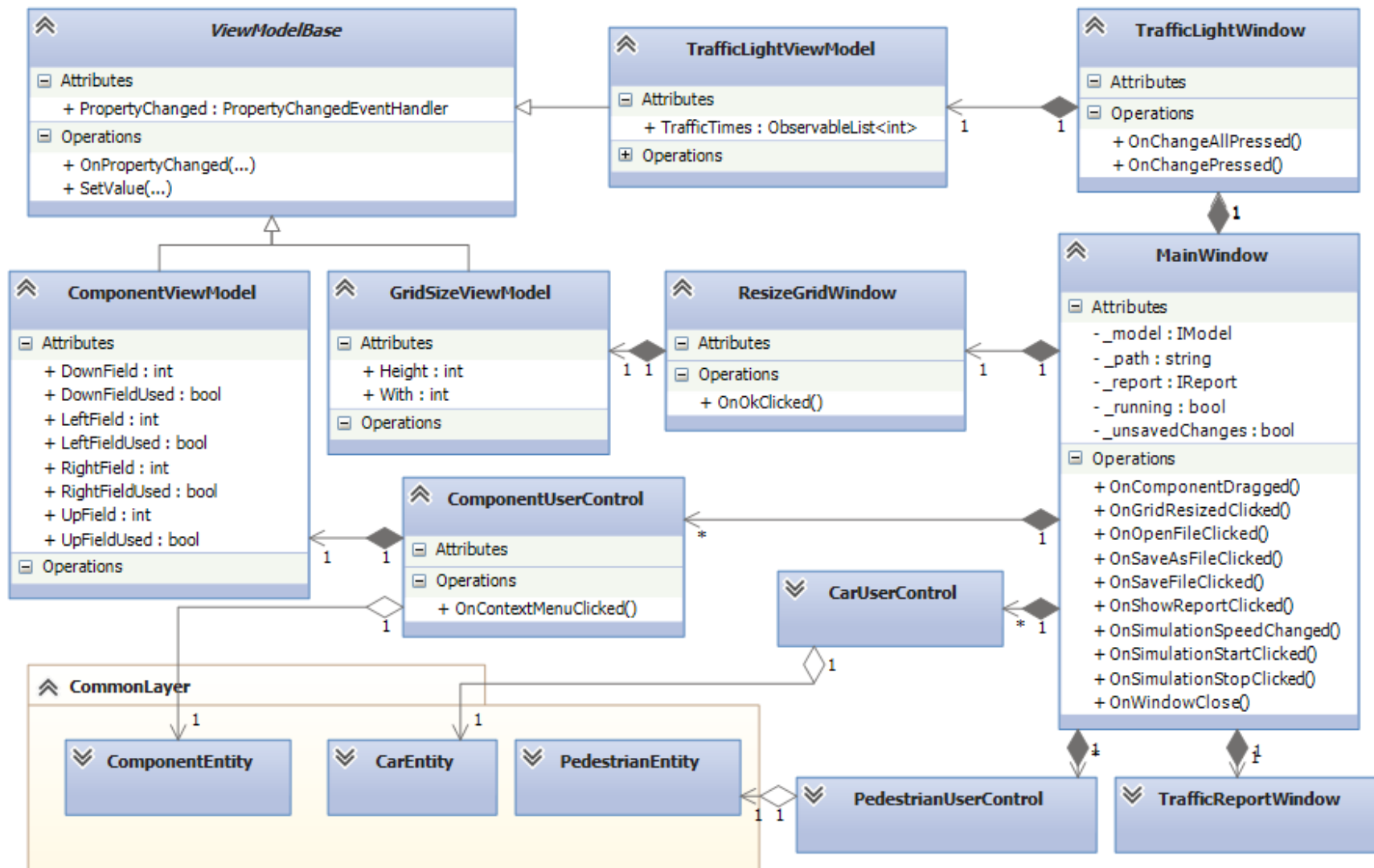


Figure 5: Class diagram view layer

### 3 Sequence diagrams

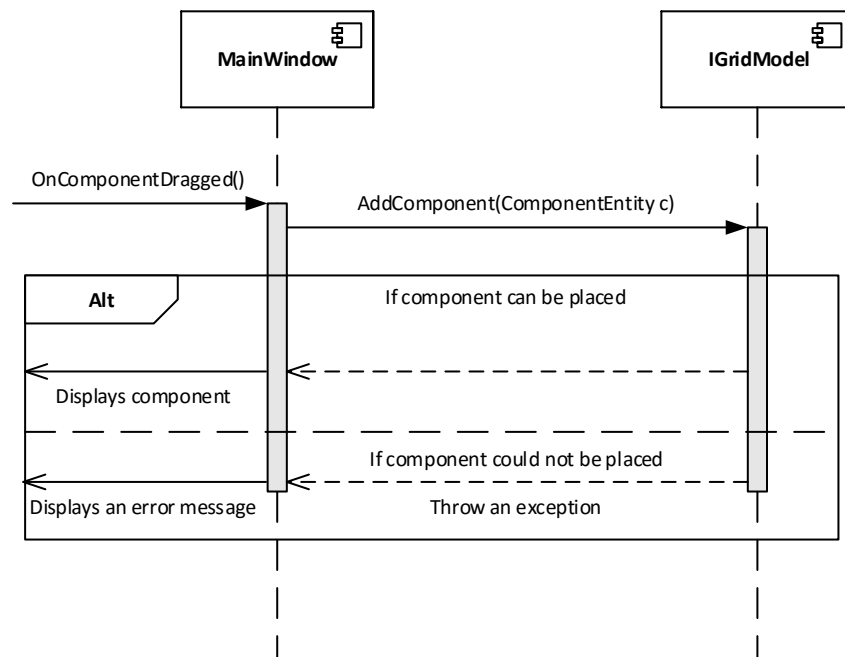


Figure 6: Positioning crossroad.

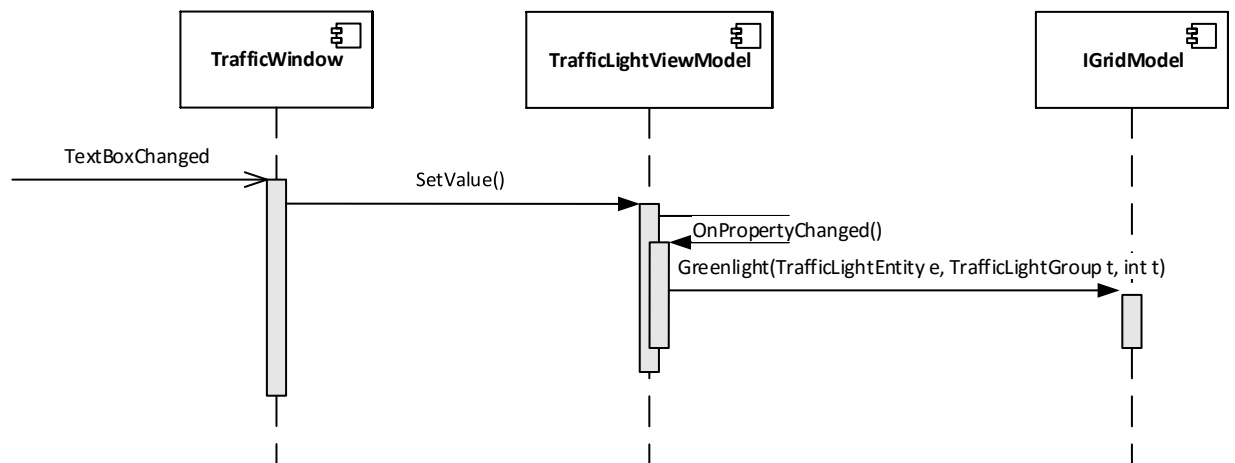


Figure 7: Configuring traffic light times.

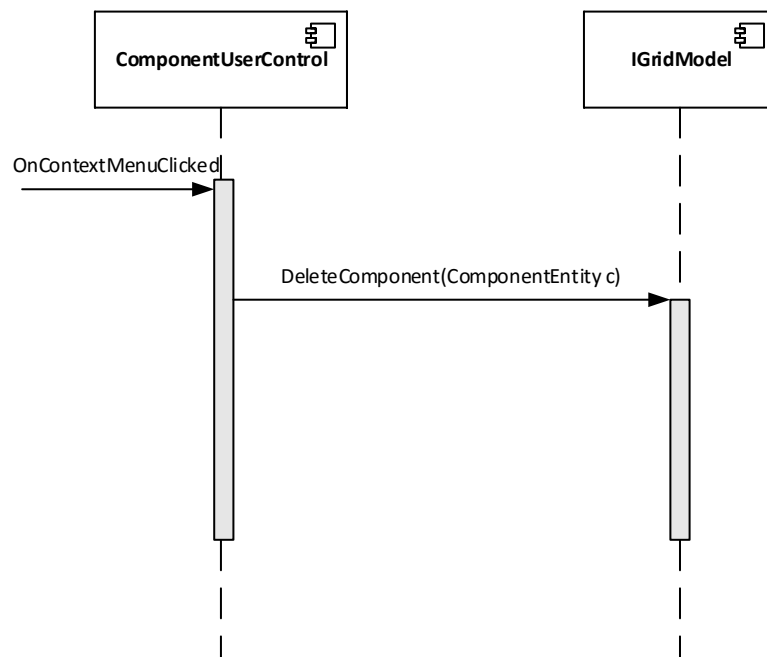


Figure 8: Deleting components.

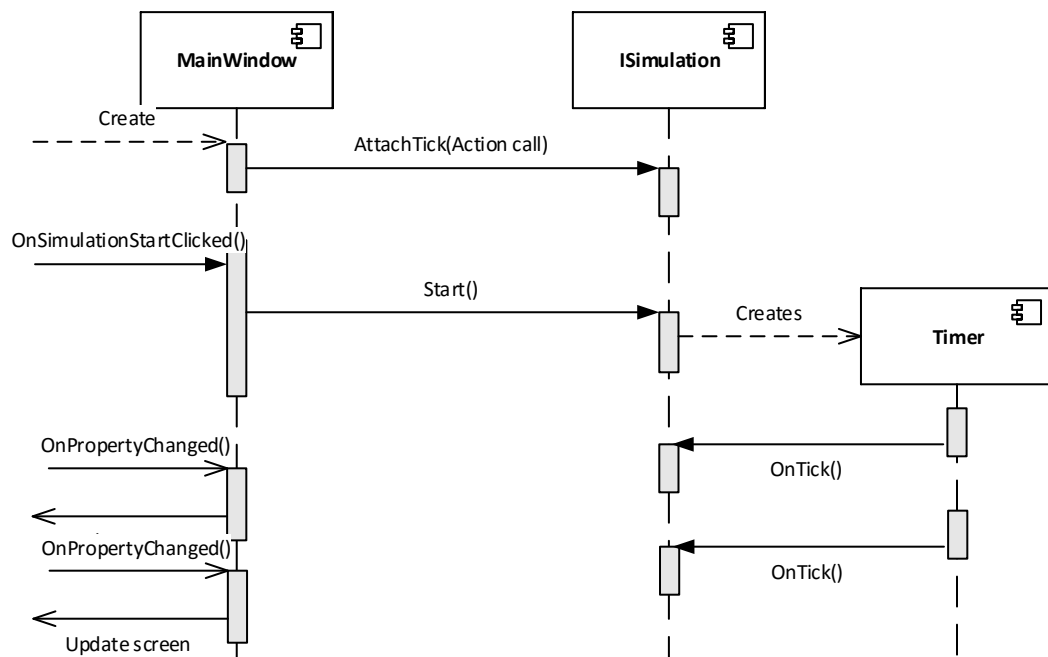


Figure 9: Running simulation.

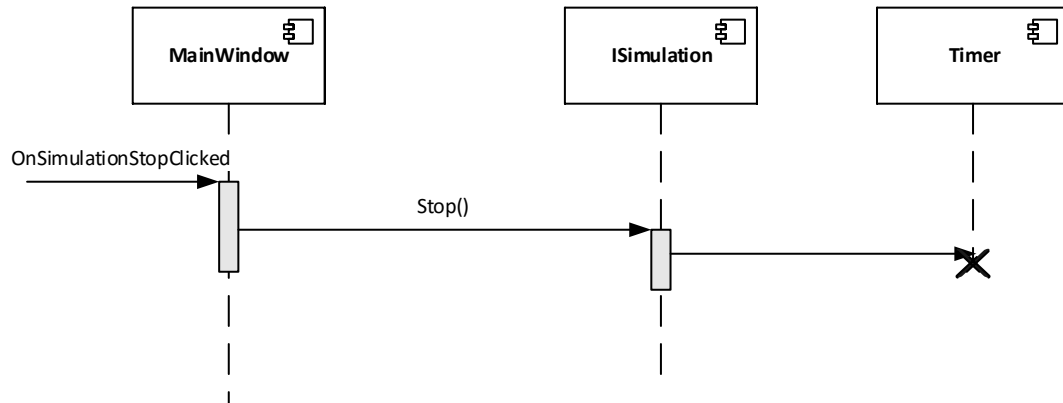


Figure 10: Stopping simulation.

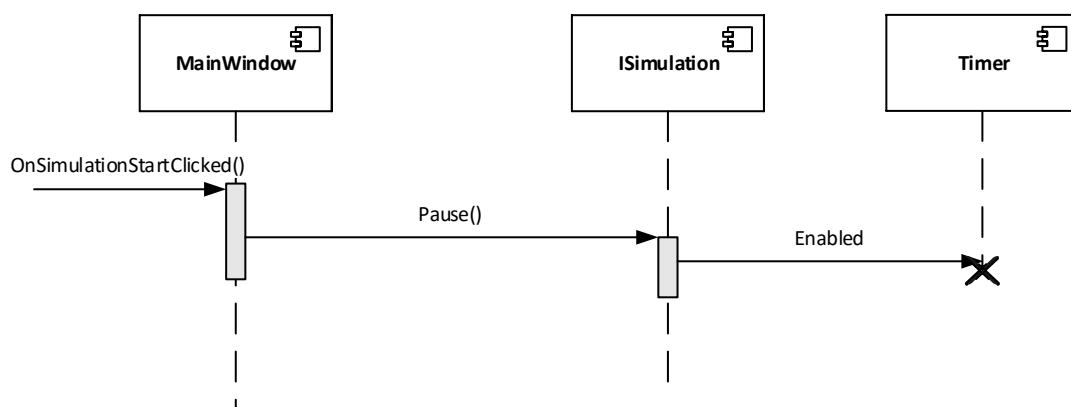


Figure 11: Pausing simulation.

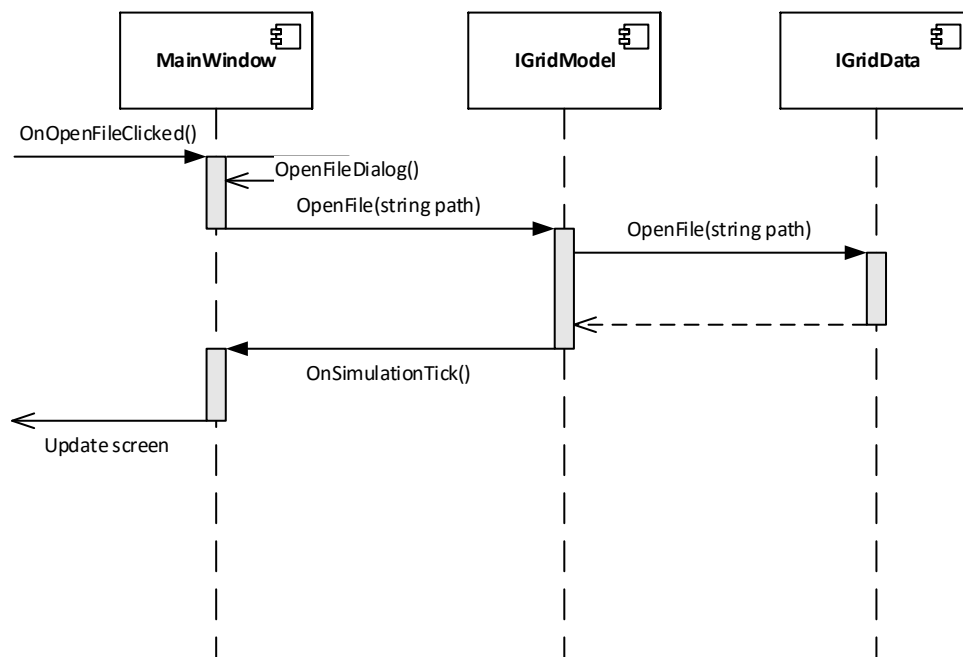


Figure 12: Opening a file.

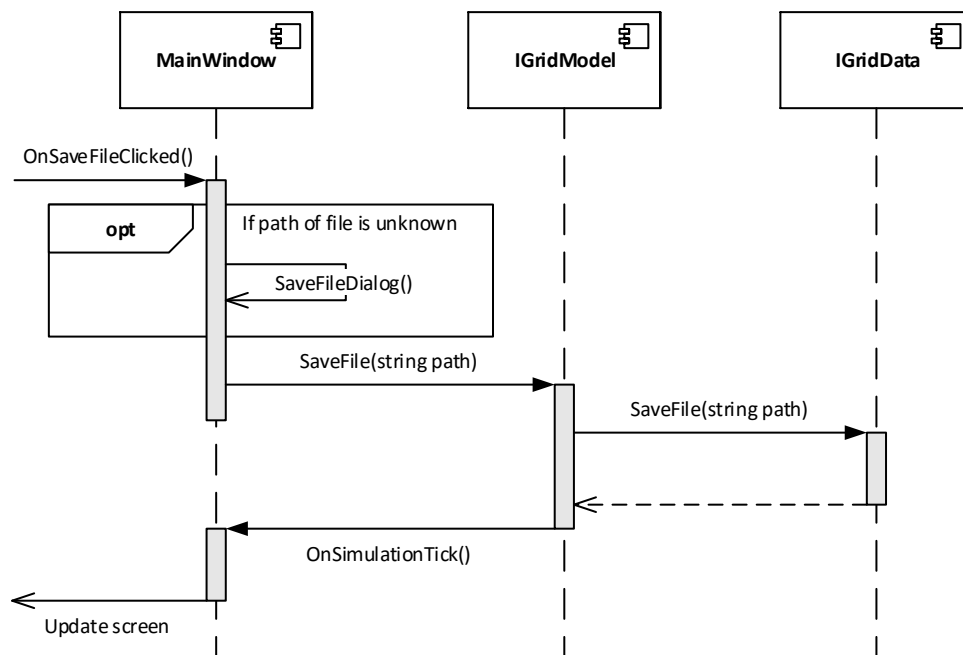


Figure 13: Saving a file.

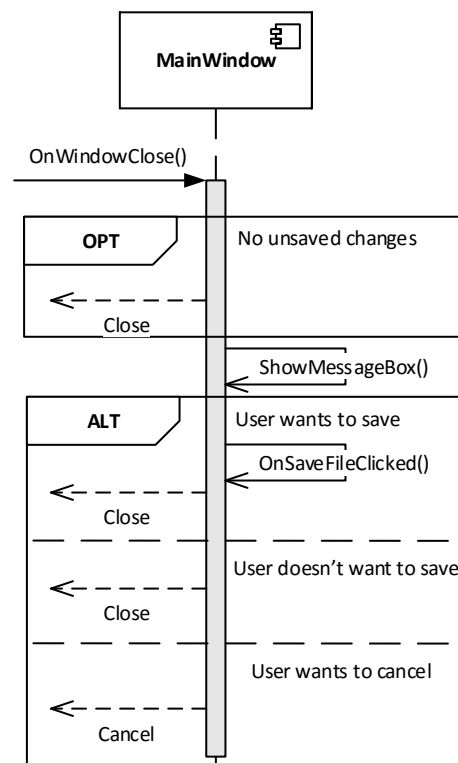


Figure 14: Close window.